Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

Пояснювальна записка

до дипломної роботи
магістр
(освітньо-кваліфікаційний рівень)

на тему: **«МОБІЛЬНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ВІЗУАЛІЗАЦІЇ АРХІТЕКТУРНИХ ПРОЕКТІВ З ВИКОРИСТАННЯМ ДОПОВНЕНОЇ РЕАЛЬНОСТІ»**

Виконав: студент 1 курсу, групи 2КІ-18м
спеціальності
123 – Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)

Кокошко Станіслав Андрійович
(прізвище та ініціали)
Керівник доц. каф. ОТ, к.т.н. Черняк О. І.
(прізвище та ініціали)
Рецензент д.т.н., проф каф. МБІС Яремчук Ю.Є
(прізвище та ініціали)

м. Вінниця - 2020 рік

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень магістр
Напрям підготовки 12 – Інформаційні технології
Спеціальність 123 – Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри Т. Б. Мартинюк

_____
"____"
_____20___року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Кокошка Станіслава Андрійовича

1. Тема проекту (роботи) мобільне програмне забезпечення для візуалізації архітектурних проектів з використанням доповненої реальності
Керівник проекту (роботи) Черняк Олександр Іванович, к.т.н., доц. каф. ОТ
затверджені наказом вищого навчального закладу від
"___"_____20__року №___
2. Строк подання студентом проекту (роботи)_____
3. Вихідні дані до проекту (роботи) список технічної літератури, аналіз, вивчення та дослідження процесів сканування області за допомогою доповненої реальності, технічне завдання на магістерську роботу.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз існуючих підходів та засобів до розв'язання задачі. Загальний огляд доповненої реальності. Огляд програмних засобів для порівняння доповненої реальності. Опис та обґрунтування інструментів для зберігання файлів. Опис та обґрунтування програм для розробки. Розробка та опис програми реалізації алгоритмів розв'язання задачі. Практична реалізація. Схема методу сканування точок. Процес сканування точок. Підрахунок та оптимізація за допомогою їх мінімізації. Створення користувацького інтерфейсу. Тестування додатку. Класи та функції програми. Економічна частина.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Блок-схема створення об'єктів, діаграма прецедентів, діаграма класів, діаграма станів, зображення діаграми послідовностей, схематичне представлення роботи методу пошуку точок, схематичне представлення роботи процесу побудови області, схематичне доповнене представлення

роботи методу сканування точок та побудова області, блок схема роботи алгоритму, тестування і перевірка програми, лістинг програми.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Технічний розділ	Черняк О. І.		
Економічний розділ	Глущенко Л. Д.		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
	Огляд існуючих підходів до розв’язання задачі		
	Аналіз існуючих підходів та засобів до розв’язання задачі		
	Узагальнення інформації		
	Розробка методу та програмної реалізації додатку доповненої реальності		
	Узагальнення інформації		
	Проведення економічних розрахунків		
	Оформлення пояснювальної записки до дипломної роботи		
	Оформлення додатків та графічного матеріалу		

Студент _____ Кокошко С.А

(підпис) (прізвище та ініціали)

Керівник роботи _____ Черняк О. І.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дана магістерська кваліфікаційна робота присвячена розробці та програмній реалізації алгоритму, який довзоляє пришвидшити сканування області камерою реального світу.

Високий рівень вирішення поставленої задачі досягнуто за рахунок використання сучасної мови програмування C#.

В даній магістерській роботі виконано дослідження і аналіз існуючих підходів та засобів до розв’язання задачі, розглянуто програмні засоби для порівняння доповненої реальності та розроблено новий метод сканування та отримання точок та їх мінімізація.

REVIEW

This master's qualification is dedicated to the development and implementation of an algorithm that accelerates scanning of both with a real-world camera.

A high level of solution to this problem has been achieved through the use of modern C # programming language.

In this master's thesis, research and analysis of existing approaches and tools for solving the problem were performed, software for comparison of augmented reality was considered, and a new method of scanning and obtaining points and minimizing them was developed.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Протягом багатьох років наука розвивалась і розвивається досить швидко для багатьох галузей діяльності. В Україні набирає популярність доповнена реальність, від чого з’являється багато іноземних компаній та інвесторів, які готові вкладати гроші у дану галузь, так як вона має широкий спектр взаємодії та є кросплатформенною. Дана технологія часто використовується у бізнес-проектах, щоб зробити візуальне представлення віртуальних об’єктів у реальному світі за допомогою телефону та його камери. Для розміщення об’єктів, на початку, виконується сканування області, що відбувається повільно або взагалі відсутнє, тому вдосконалення цієї процедури (сканування області) є актуальним завданням.

Вираховування області відбувається за допомогою звичайних математичних обчислень, що є не завжди раціональним та робить сканування області телефоном повільним.

Матриці та їх обчислення надає можливість швидше обрахувати вхідні дані та отримати точки, які повертають певну скановану область взаємодії, що відображається у додатку.

Предметом даної магістерської роботи є програмна реалізація додатку для розміщення об’єктів на відсканованій області за допомогою доповненої реальності, що дозволяє пришвидшити роботу додатку та обрахунку точок для створення цієї області.

Тема магістерської роботи є актуальною, тому що існує потреба у використані даної технології малими та великими компаніями, тому удосконалення додатковими алгоритмами, а саме обчислення точок матрицями. Запропонований метод та програмна реалізація зменшить час сканування та обробки області та будуть корисними для даних компаній.

Мета та завдання дослідження. Метою дослідження магістерської роботи є пришвидшення сканування області розміщення об’єктів у доповненій реальності.

Задля досягнення поставленої мети у роботі необхідно розв’язати наступні задачі:

– провести аналіз сучасних методів та засобів сканування області в доповненій реальності;

– розглянути існуючі методи вирішення задачі сканування області та обрати й обґрунтувати вибір методу, що задовільнив би мету даної магістерської роботи;

– запропонувати швидкодіючі алгоритми методу обчислення матриці точок;

– виконати програмну реалізацію розроблених алгоритмів;

– провести тестування програмного продукту.

Об’єкт дослідження – сканування точок поверхні у доповненій реальності.

Предмет дослідження – методи та програмні засоби обчислення точок та відображення їх у додатку задля розміщення на них об’єктів.

Методи дослідження. Методи теорії алгоритмів, технологія об’єктно-орієнтованого програмування технологія доповненої реальності Google.

Наукова новизна одержаних результатів полягає в тому, що покращено метод сканування області для розміщення об’єктів, який на відміну від існуючого дозволяє аналізувати меншу кількість точок за рахунок розробки нових алгоритмів, що призводить до підвищення швидкості сканування поверхні.

Практичне значення одержаних результатів. Розроблено нові більш швидкі алгоритми для доповненої реальності, які реалізовані засобами мови програмування C# за допомогою програмної платформи Unity. Виготовлений програмний продукт практично готовий до комерційного використання.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням методів та інформаційних технологій під час проведення наукових досліджень, строгим виведенням аналітичних співвідношень, порівнянням результатів з відомими, та збіжністю результатів дослідження з результатами, що отримані під час тестування розробленого програмного засобу.

Особистий внесок магістранта. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно.

Апробація одержаних результатів. Основні результати отримані в процесі дослідження за темою даної магістерської роботи, відображені в тезах, що були подані на наукову конференцію «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ-2019».

1 АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ТА ЗАСОБІВ ДО РОЗВ’ЯЗАННЯ ЗАДАЧІ

1.1. Загальний огляд доповненої реальності

Що таке доповнена реальність? Доповнена реальність є результатом використання технології для накладання інформації - звуків, зображень та тексту - на світ, який ми бачимо.
В чому різниця між доповненою реальністю та віртуальною реальністю? Віртуальна реальність означає, що ви створюєте комп’ютерні середовища для взаємодії з вами та занурення в них. Доповнена реальність (також відома як AR) додає до реальності, яку ви зазвичай бачите, а не заміняючи її.
Доповнена реальність у сучасному світі. Доповнену реальність часто представляють як своєрідну футуристичну технологію, але її форма існує вже роками. Наприклад, головні дисплеї у багатьох винищувальних літаках ще в 90-х роках показували б інформацію про відношення, напрямок і швидкість літака, і лише через кілька років вони могли показати, які об'єкти в полі зору були цілями .
За останнє десятиліття різні лабораторії та компанії створили пристрої, які дають нам доповнену реальність. У 2009 році група рідинних інтерфейсів лабораторії MIT Media представила SixthSense - пристрій, який поєднував використання камери, невеликого проектора, смартфона та дзеркала. Пристрій висить на грудях користувача шнурово з шиї. Чотири сенсорні пристрої на пальцях користувача можуть використовуватися для маніпулювання зображеннями, запроектованими SixthSense.
Google розгорнув Google Glass у 2013 році, перемістивши розширену реальність до більш зношеного інтерфейсу; в цьому випадку окуляри. Він відображається на екрані об'єктива користувача через невеликий проектор і реагує на голосові команди, накладаючи зображення, відео та звуки на екран. Google витягнув Google Glass наприкінці грудня 2015 року.
Як це буває, телефони та планшети - це спосіб доповненої реальності потрапляти в життя більшості людей. Наприклад, додаток Vito Technology Star Walk дозволяє користувачеві орієнтувати камеру у своєму планшеті чи телефоні в небо і бачити назви зірок і планет, накладених на зображення. Інший додаток під назвою Layar використовує GPS смартфона та його камеру для збору інформації про оточення користувача. Потім він відображає інформацію про ресторани, магазини та визначні місця поблизу.
Деякі програми для планшетів та телефонів також працюють з іншими об'єктами. Компанія Disney Research розробила книжку розмальовок AR, в якій ви розфарбовуєте персонаж у звичайній книзі (хоча і сумісному з додатком) та запускаєте програму на пристрої. Додаток звертається до камери та використовує її для виявлення того, який персонаж ви забарвлюєте, і використовує програмне забезпечення для відновлення персонажа в 3D-символі на екрані.
Один з найпопулярніших способів проникнення AR в повсякденне життя - це через мобільні ігри. У 2016 році гра AR «Pokémon Go» стала сенсацією у всьому світі, а понад 100 мільйонів оцінювали користувачів на піку, повідомляє CNET. За підсумками Forbes, він заробив понад 2 мільярди

доларів і порахував. Гра дозволила користувачам бачити персонажів покемонів, що підстрибують у власному місті. Метою було захопити цих кишенькових монстрів, а потім використовувати їх для боротьби з іншими, локально, в тренажерних залах AR.
У 2018 році "Гаррі Поттер: Таємниця Хогвартса" стала мобільною ігровою сенсацією AR. Гра дозволяє користувачам бачити навколишній світ Хогвартсу, маючи можливість кидати заклинання, використовувати зілля та вчитися у вчителів Хогвартса. На момент написання цієї гри в магазині Google Play було завантажено близько 10 мільйонів завантажень.
Дослідники також розробляють голограми, які можуть зробити VR ще на крок, оскільки голограми можуть бачити і чути натовп людей відразу. "Хоча дослідження з голографії відіграють важливу роль у розробці футуристичних дисплеїв та приладів доповненої реальності, сьогодні ми працюємо над багатьма іншими додатками, такими як ультратонкі та легкі оптичні пристрої для камер та супутників", - дослідник Лей Ванг, докторант Наукова школа фізики та інженерії ANU, йдеться у повідомленні.
Майбутнє доповненої реальності. Це не означає, що телефони та планшети будуть єдиним місцем проведення AP. Продовжуються дослідження щодо включення функціональності AR в контактні лінзи та інших носячих пристроїв. Кінцевою метою доповненої реальності є створення зручного та природного занурення, тому є відчуття, що телефони та планшети будуть замінені, хоча не зрозуміло, якими будуть ці заміни. Навіть окуляри можуть набути нової форми, оскільки «розумні окуляри» розроблені для незрячих.
Як і будь-яка нова технологія, AR має багато політичних та етичних питань. Наприклад, Google Glass викликав занепокоєння щодо конфіденційності. Деякі стурбовані тим, що розмови можуть бути записані назовні, або знімки, або вважають, що їх можна визначити за допомогою програмного забезпечення для розпізнавання облич. AR окуляри, контакти та багато іншого, як Glass-X та Google Lens, хоча йдуть уперед у виробництві та продажах.

1.2. Огляд програмних засобів для порівняння доповненої реальності

Щоб задоввільнити потреби користувача, на ринку програмного забезпечення знаходиться досить багато додатків для здійснення сканування області та різною маніпуляцією з нею, такою як розмістити об’єкт, намалювати рисунок та безліч іншого функціоналу. Серед таких рішень є платні додатки, додатки які не відскановують область та додатки які виконують сканування досить повільно, що може принести доскомфорт у використанні користувачу та додатки які інсують тільки на певних платформах, таких як iOS. Для прикладу будуть взяті такі додатки як Houzz, IKEA Plcae, Hutch, DecorMatters, Homestyler Interior Design.
Дані додатки забезпечують процеси розташування об’єктів на сцені та є зручні в плані інтерфейсу, можуть підтягувати об’єкти з бази даниз або мати локаньку базу, розташовану у ресурсах додатку.
Houzz – додаток для проектування, побудови та декорування. Завантажити його можна Play Market.
Плюси додатку полягають у наступному:

- має велику базу об’єктів;
- має можливість сфотографувати те, що розміщено на сцені;
- дає можливість обмінюватись картинками;
- має можливість підключення до соціальних мереж.

Мінуси додатку:

- обов’язкова реєстрація;
- моделі можуть бути платними;
- сканування області відбувається повільно.

Інтерфейс даного додатку ділиться на нижню панель з переходом між сторінками та має дещо громіздкий вид.
Наступний додаток – IKEA Place – додаток меблевого гіганту, що надає можливість розміщення будь-якого елементу з їхнього магазину.
Плюси даного додатку наступні:

- розміщення результату у соціальних мережах;
- велика база власних меблів;
- відображає предмети в реальному розмірі.

Мінуси додатку:

- знаходиться тільки власна продукція, немає можливості додати свої об’єкти;
- немає можливості швидкого сканування області;
- обов’язкова реєстрація користувача;

– доступний додаток тільки на iOS у Apple Store.

Інтерфейс додатку дещо відрізняється від попереднього програмного рішення, він має менше функціоналу, але він зрозумілий користувачу, від чого не виникатиме дискомфорту при користуванні.

DecorMatters – допомагає візуалізувати будь-який дизайн інтрер'єру.
Плюси даного додатку:

– відносно нова розробка;

– має більшу кількість розміщуваних об'єктів;

– додаток безкоштовний.

Мінуси даного додатку:

– платні моделі;

– немає можливості зберігти зображення;

– доступний тільки на Apple Store;

– відстуня можливість швидкого сканування області.

Homestyler Interior Design – забеспечує точний та простий у взаємодії веб-конструктор, що з часом з'явилась і на мобільному ринку з доповненої реальністю.
Плюси даного додатку:

– зручний інтерфейс;

– можливість синхронізувати візуалізацію між мобільну та веб версіями;

– доступність;

До мінсуів відносяться:

– мала база об'єктів;

– повільне сканування області;

– обов'язкова реєстрація.

Інтерфейс даної програми не розрахований на портретну орієнтацію телефону, що є не зовсім зручним враховуючі останні денденції в озробці смартфонів та додання до них великих діагоналей екрану.
Hutch – орієнтований на просту ідею дизайну інтрер'єру, допомагає налаштувати сцену з об'єктами не роблячи зайвих рухів та не додаючи зайвих каталогів.
Плюси:

– має велику базу;

– надає можливість розмістити об'єкти на сцені.

Мінуси:

– обов'язкова реєстрація;

- відсутність збереженя виду сцени;
- доступна тільки в Apple Store;
- візуалізація у вже готових кімнатах.

Порівняльна таблиця 1.1 приведена нижче.

Таблиця 1.1 – Порівняльна характеристика програм

	Можливість швидкого сканування області	Розміщення об'єктів на сцені	Завантаження вже існуючих об'єктів з бази даних	Зручний, не громіздкий інтерфейс	Адаптація під діагоналі телефону під певне співвідношення сторін	Доступність для безкоштовного використання
Houzz	-	+	-	-	+	-
IKEA Place	-	+	-	+	-	+
DecorMatters	+	+	-	-	+	+
Homestyler Interior Design	-	+	+	+	+	-
Hutch	+	+	-	-	-	-
Нова розробка	+	+	+	+	+	+

1.3. Порівняльний аналіз інструментів для зберігання файлів

У ході розробки мобільного додатку потреба у зберіганні ассетів програми, в яких зберігаються префаби для сцени. Порівняння було проведено міжи таким засобами як AWS, Azure та Firebase.

AWS та його плюси:

– Comprehensive (Всебічний) - перехід із сховища на місці до хмарного просто за допомогою AWS завдяки інвестиціям, які вони зробили в освіту та навчання. Хмарна електростанція пропонує безліч знань на своєму сайті, включаючи документацію та навчальні посібники для початку роботи з AWS, їх різноманітними послугами тощо. AWS також має Партнерську мережу, яку складають професійні фірми, які допомагають клієнтам проектувати, архітектувати, будувати, мігрувати та керувати їх робочим навантаженням та додатками на AWS. Щоб отримати статус нашого консалтингового партнера, співробітники компанії Lofty Labs

пройшли кілька акредитацій та сертифікацій, отримали довідки клієнтів про AWS та підтвердили дохід завдяки послугам AWS;

– Cost-Effective (Економічно-ефективний) - Незалежно від того, ви стартап або велика корпоративна компанія, ви заощадите, лише використовуючи послуги, потрібні вашому бізнесу в будь-який момент. AWS має конкурентоспроможне ціноутворення, яке є часткою витрат, які коштують локальні рішення. Ви можете порівняти вартість запуску ваших додатків у локальному або колокаційному середовищі з AWS за допомогою цього зручного інструменту обчислення;

– Flexible (Гнучкий) - Незалежно від того, чи переходите ви в хмару вперше чи переходите з іншої хмарної служби, AWS має всі ресурси, необхідні для оптимізації вашого I.T. інфраструктура. Їх модель підтримує масштабування ресурсів вгору або вниз, а це означає, що ваш бізнес не повинен турбуватися, коли проблема є потужністю або коли коливання потреби змінюються;

– Security (Безпечний) - Захист вашого бізнесу від потенційних злому даних та витоків є важливим пріоритетом для AWS. Вони мають багато визнаних сертифікатів відповідності та дотримуються законів про конфіденційність з усього світу. Nasdaq, Dow Jones та HealthCare.Gov використовують AWS, що є свідченням того, наскільки AWS є надійною хмарною службою;

– Enhanced Productivity (Підвищена продуктивність) - надання AWS підтримки хмарних обчислень означає усунення обов'язків та ризиків, пов'язаних із внутрішнім I.T. інфраструктура. Це також зменшує потребу в I.T. обслуговуючий персонал та економить час та гроші вашої компанії в довгостроковій перспективі;

– Innovative (інноваційний) - є статті, які стверджують, що відданість Amazon інноваціям, а не їх конкурентоспроможні ціни, в кінцевому рахунку перемагає своїх клієнтів. Не дивлячись на цінову війну з Microsoft та Google, Amazon ще не стикається з конкуренцією у своїй відданості винаходові та експериментам. За словами учасника останньої

конференції AWS про: Invent Conference 2016 року, AWS запустила майже 1000 нових послуг минулого року;

– Global Leader (Глобальний лідер) - веб-сервіси Amazon працюють у 190 країнах та підтримують понад мільйон активних клієнтів. Вони зараховують кілька найбільших і найменших підприємств у світі як своїх клієнтів і навіть обслуговують державний сектор.

Azure та його плюси:

– Apps management - IaaS надає перевагу організації створювати, розгортати та керувати програмами швидко та простіше. Організація може запустити веб-сайт або створити веб-додаток та підтримувати інфраструктуру, налаштовуючи хмарне програмне забезпечення;

– Flexibility - Azure забезпечує відчутний рівень гнучкості, що дає вам можливість функціонувати за потребою. Ви можете платити, як споживаєте, переходите на Azure, пристосовуєтесь до коливань бізнесу і т. Д. Таким чином, про інфраструктуру весь час не турбуєтесь;

– Agility - Azure швидкий з точки зору розгортання, експлуатації та масштабованості. Це дає конкурентну перевагу компаніям, які приймають Azure. Будучи найсучаснішою хмарною технологією, інфраструктуру та додатки можна зробити гнучкими;

– Compliance - дані, що зберігаються, відповідають нормам, що дуже корисно, особливо для юридичного та фінансового секторів. Він повністю побудований на вимогах безпеки та конфіденційності, щоб будь-який бізнес охоче взявся за справу;

– Storage - відомо, що Azure має кілька центрів обробки даних та пунктів доставки. Це сприяє більш швидкій доставці вмісту, оптимальному користувальницькому використанню, збереженню будь-яких даних, а також дозволяє обмінюватися даними на віртуальних машинах за потребою з надійною та швидкою швидкістю;

– Security - дані про Azure захищені середовищем шпигунських фільмів. У центрах обробки даних є дворівнева аутентифікація, зчитувачі доступу до проксі-карт, ручні геометричні зчитувачі геометрії, глобальна

команда реагування на інциденти. Таким чином злом зменшується більшою мірою;

– Analytics Support - Azure має вбудовану підтримку для аналізу даних та отримання уявлень, які допомагають в керованих службах SQL, машинному навчанні, потоці та Cortana Analytics. Таким чином, ваші бізнес-рішення приймаються розумнішими, що веде до нових можливостей;

– Unified Delivery Plan - Azure забезпечує цілісне рішення. Усі інструменти для управління джерелом, тестування блоків, доставки, інструментів живого перегляду або тестування інтеграції доступні під однією парасолькою. Це призводить до відсутності страху перед питаннями, пов'язаними з інтеграцією та наступністю;

– Disaster Recovery - весь час перебування в Інтернеті забезпечує довіру клієнтів / користувачів. Можливості відновлення аварійних ситуацій Azure, як регіональні / глобальні варіанти відмови; перезавантаження, а режими очікування та гарячого / холодного стану підтримують захист від катастроф;

– Deployed Anywhere - приймаючи підхід до гібридної моделі клієнтів, Microsoft пропонує пакет Azure, який пропонує всі переваги для них. Це дозволяє бізнесу легко вибирати сховище даних та будь-який пов'язаний з цим перехід;

– Updates - у Azure оновлення програмного забезпечення автоматизовані. Більше того, він оновлюється в режимі реального часу, щоб інфраструктура та додатки були пристосовані до дня. Таким чином, це допомагає бізнесу швидше розвиватися з меншим впливом вуглецю. Чудовий спосіб почати та реалізувати цю перевагу - стати експертом у цій галузі. Можливо, ви вже працювали / працювали над цими рішеннями та набули досвіду роботи. Тим не менш, якщо ви можете підтвердити свої знання та вміння, це ще більше розширює вашу кар’єру.

Firebase та його плюси:

– Real-time Database - база даних у режимі реального часу - це хмарна база даних. Дані зберігаються як JSON і синхронізуються постійно з

кожним пов'язаним клієнтом. Коли ви створюєте крос-платформні програми з пакетами SDK для iOS, Android та JavaScript, більша частина попиту ваших клієнтів базується на одному екземплярі бази даних в реальному часі і, отже, отримує оновлення з найновішими даними. Використовуючи цю функцію Firebase, немає необхідності створювати власну базу даних або власний API, Firebase обробляє всі компоненти, які зазвичай поставляються разом зі створенням бекенда для додатків. Він дає адаптовану мову, засновану на виразах, для визначення того, як мають бути організовані ваші дані та коли інформація може бути використана з або складена;

– Hosting - хостинг - це веб-контент, який відповідає виробництву. За допомогою хостингу ви можете швидко та ефективно надсилати веб-програми та статичний вміст до мережі доставки вмісту (CDN) за допомогою однієї команди. Це дуже простий процес у Firebase, оскільки він містить користувальницьку підтримку домену, глобальну CDN та автоматичну підтримку SSL-сертифікатів для цього. Незалежно від того, чи надсилаєте ви просту цільову сторінку програми або складний прогресивний веб-додаток. Хостинг надає вам інфраструктуру, функції та інструменти, призначені для передачі та керування статичними веб-сайтами;

– Authentication - перевірка автентичності Firebase надає сервісні сервіси, прості у використанні SDK та миттєві бібліотеки інтерфейсу для підтвердження клієнтів у вашій програмі. Він підтримує автентифікацію за допомогою паролів, ідентифікатора електронної пошти або імені користувача. Ви можете дозволити користувачам входити у ваш додаток Firebase або використовуючи FirebaseUI як повне рішення для аутентифікації, що випадає, або використовуючи пакет SDK Firebase Authentication, щоб вручну інтегрувати один або кілька методів входу у вашу програму;

– Storage - він створений для розробників додатків, яким потрібно зберігати та обслуговувати створений користувачем вміст, наприклад фотографії чи відео. Це забезпечує безпечну передачу документів та завантаження програм Firebase, незалежно від якості мережі. Ви можете

використовувати його для зберігання зображень, звуку, відео чи іншого вмісту, створеного користувачем. Firebase Storage підтримується Google Cloud Storage, здатним, базовим та економічно ефективним сервісом зберігання об’єктів;

– Cloud Messaging - це крос-платформене обмін повідомленнями, яке дозволяє вам надійно передавати повідомлення за нульовий рахунок. Ви можете повідомити клієнта, що нова електронна пошта чи інша інформація доступна для синхронізації. Ви можете надсилати повідомлення-сповіщення для залучення та відновлення користувачів;

– Remote Config - це хмарний сервіс, який дає вам можливість змінити поведінку та зовнішній вигляд вашої програми, не вимагаючи від користувачів завантаження оновлення програми. Ваша програма контролює, коли застосовуються оновлення, і вона може якомога частіше перевіряти наявність оновлень і застосовувати їх, маючи незначний вплив на виконання;

– Test Lab - тестова лабораторія використовується для тестування вашої програми на гаджетах, розміщених у центрі обробки даних Google. Це допомагає вам знайти проблеми, які трапляються лише у певних конфігураціях гаджетів. Результат тесту включає журнали, відео та скріншоти, які доступні у вашому проекті на консолі Firebase. Навіть якщо ви не склали жодного тестового коду для своєї програми, Test Lab може, відповідно, практикувати вашу програму, шукаючи збої;

– Crash Reporting - це допомагає створювати докладні звіти про помилки, які збираються в групи порівняльного потоку стека, що викликається серйозністю впливу на користувачів. Окрім автоматичних звітів, ви можете реєструвати власні події, щоб допомогти захопити кроки, що призводить до збоїв;

– Notifications - це безкоштовна послуга Firebase, яка дає змогу зосереджувати сповіщення користувачів для розробників мобільних додатків. Це дає вибір розробникам та організаціям, які шукають адаптовану

платформу сповіщень, яка вимагає мінімальних зусиль для кодування, і графічну консоль для надсилання повідомлень.

Після порівняння особливостей та можливостей сервісів для зберігання файлів було обраний сервіс Firebase через його можливості та безкошовне використання.

Виходячи з вище наведених прикладів можна скласти порівняльну таблицю з висновками який із сервісів підходить краще, і після цього порівняння слід використовувати інструмент Firebase. Таблиця 1.6 показана нижче.

Таблиця 1.2 – Порівняльна характеристика засобів зберігання даних

	Real-time database	Notitfications	Crash reporting	Test lab	Remote config	Storage
AWS	+	-	-	-	+	-
Azure	+	+	+	-	+	+
Firebase	+	+	+	+	+	+

1.4. Порівняльний аналіз програмних засобів для розробки

Одне з найефективніших застосувань AR дотепер було в бізнес індустрії[6]. Для створення ігор у віртуальній реальності можна використати одну з нижче наведених програм:

– CryEngine;

– Unreal Engine;

– LibGDX;

– AppGameKit VR;

– Unity3D.

AppGameKit є легким для вивчення початківцями матеріалу та досить потужним для ветеранів у створенні захоплюючих ігор.

– розгортає ігри на IPhone та IPad, телефони на планшети на ОС Android, настільки комп’ютери, ноутбуки та планшети Windows, пристрої Mac OS, Linux, HTML5 та інші;

– дозволяє швидко і легко створювати та продавати ігри;

– дає можливість вивчати кодування за допомогою експертних керівництв та підручників.

AppGameKit Virtual Reality Kit

– довзоляє виконувати діагностичні перевірки обладнання та SteamVR;

– дозволяє надати користувачу досвід у грі як сидячи, так і стоячи;

– встановлює масштаб і діапазон камери;

– дозволяє контролювати положення та обертання гравця у 3D-сцені;

– визначає рух як лівої, так і правої руки користувача та передаватиме вихідні дані за допомогою джойстика.

За користування такою комфортністю у використанні движка потрібно заплатити 80 долларів.

LibGDX - це Java фреймворк, який надає крос-платформне API для розробки ігор і додатків, що працюють в режимі реального часу[14]. Це високопродуктивний, кросплатформенний ігровий фреймворк, що в першу чергу використовується для написання ігрових двигунів та ігор. Позиціонується, як фреймворк та дозволяє нам максимально зосередитися на міцному фундаменті, замість того, щоб намагатися реалізувати найновіше і найкраще з ігрових двигунів. LibGDX надає Вам гнучкість і дозволяє уникнути суворої методології. За допомогою даної бібліотеки, можна використовувати один і той же код як для систем настільних комп'ютерів так і мобільних систем. Бібліотека є кросплатформенною і підтримує Windows, Linux, Mac OS X, Android, iOS, та браузери з підтримкою WebGL.

Недоліком розробки ігор на цьому дживку є те, що він не надає зручного інтерфейсу для спрощеного користування ним, реалізація відбувається на мові програмування Java, не підтримуючи інші.

Crytek зробив подарунок розробникам ігор, оголосивши, що CryEngine V випускається з бізнес-моделлю «Pay What You Want», що означає «плати скільки хочеш». Це значить, що розробники можуть отримати програму безкоштовно і не зазнавати особливих витрат. Тим не менш, ті хто хоче платити за використання двигуна, побачать, що їх гроші перейдуть у розвиток нової інді-фонд компанії, який був створений для підтримки розробників інді ігор, які знаходяться в різних кутках світу.

Останній движок Crytek також представляє Cryengine Marketplace, місце, де розробники можуть охопити окремі гілки розділу аксесуарів, створену компанією Crytek разом з іншими елементами створеними

спільнотою та «надійними постачальниками». До таких елементів входять 3D-об’єкти, звуки та матеріали такі, як житлові будинки, вокзал тощо.

CryEngine V включає в себе новий sandbox редактор, новий API для підтримки C#-скриптів, підтримку DirectX 12, перероблений низькорівневий рендеринг, розширену хмарну систему, нову систему партиклів (частинок), поліпшений профайлер, а також підтримка Visual Studio 2015. Редактор рівнів був теж перебудований за допомогою краще організованих меню та панель інструментів з покращеними комбінаціями клавіш, що полегшують роботу у редакторі.

Unreal, продукт Epic Games, це ігровий движок, що пояснюється тим, що він стоїть за більшістю високотехнологічних ігор, які мають виняткову графіку та функції в цілому. Він має найдовшу історію, що є найкращим варіантом для тривимірних ігор для консольних і ПК ігрових платформ.
Остання версія Unreal Engine двигуна Unreal Engine 4, це, мабуть, найекономічніший варіант для шанувальників гри лише за 19 доларів на місяць та 5% доходу, а також забезпечує доступ до всіх функцій та інструментів двигуна.
Підтримуючи лише мову розробки C ++, Unreal Engine - це ігрова платформа, що надає неймовірну продуктивність в іграх. Технологія розробки включає систему візуальних сценаріїв Blueprint, машину на базі Matinee, анімаційну систему Persona та кінематографічну систему. Вона має найбільшу спільноту розробників, яка надає найкращу підтримку всім і всім, хто ризикував іграх у віртуальній реальності. Регулярний механізм оновлення гарантує постійний догляд та відповідь на кожне окреме питання, яке будь-хто може мати в межах спільноти.
Unreal Engine не тільки підтримує всі основні операційні платформи для мобільних пристроїв та ПК, він підтримує Linux, Oculus, PS4, SteamOS і Xbox One. Повне розширення вихідного коду, налаштування та виправлення помилок Unreal Engine має найновіші засоби розробки, що охоплюють кожен аспект тривимірних ігор.
Таблиця 1.3 – Порівняльна характеристика програмних засобів для розробки

	Доступність	Можливість створювати додатки з вироистанням доповненої реальності	Кросплаформерність	Підтримка мови програмування C#	Обширність спільноти	Постійна підтримка та додавання нових функцій
CryEngine	+	-	-	+	+	+
Unreal Engine	+	+	-	-	+	+
LibGDX	+	-	+	-	-	-
AppGameKit VR	+/-	-	-	+	-	-
Unity3D	+	+	+	+	+	+

Дізнавшись про Unity3D та зрозумівши основи розробки додатків стає із задоволенням розробляти додатки[9]. Далі буде представлено п’ять цікавих речей про Unity3d:

– Unity дає можливість розробляти додатки доповненої та віртуальної реальності для телефонів наступного покоління – це досить загальновідомо, але існують і інші бібліотеки та структури, такі як Argon.js або WebVR, які можуть заплутати нових користувачів. Однак ці бібліотеки обмежені на даний момент часу і насправді не дозволяють вам робити вражаючий, незрозумілий досвід, який ви, можливо, відчували, наприклад, VR ігри для Oculus або HTC Vive, або голографічні додатки для Hololens;

– зробив код один раз, а білд використовується багато разів – працюючи над проектом з Unity3D, надається можливість писати код як в C #, так і в JavaScript. Після того, що було створено, є можливість побудувати власний проект для роботи на майже всіх основних мобільних платформах [20].

– відмінний вбудований емулятор – емулятори доступні для тестування додатків для Android (Android Studio та GenyMotion) та додатків для iOS (лише для MacOS), але вбудований емулятор Unity3D дає можливість тестувати дві платформи. Ще один чудовий інструмент, де можна запустити

свій проект в емуляторі, а також маніпулювати чи внести зміни в код в режимі реального часу. Це одна з цих особливостей, яка є величезною заощаджувачем часу і дозволяє швидко створити прототипи нових матеріалів.

– Asset Store – на додаток до безлічі онлайнових підручників та дивовижних репозиторіїв на Github, Unity3D має власний Asset Store (Магазин Аксесуарів), що має величезний спектр безкоштовних і заощаджених ресурсів, таких як OpenCV Integration, LogViewer (консоль додатків для налагодження та входу в реальний пристрій), 3D-моделі з звуками та анімаціями та багато іншого.

– Cloud Build та аналітика – є одними з найяскравіших служб, випущених в 2016 році. Він автоматично збирає, розгортає та тестує додаток, для швидкого та послідовного опрацювання додатку командою, є багатоплатформерним і дозволяє безперебійне розгортання. Unity Analytics - це ще одна з декількох чудових сервісів, що не потребують додаткових SDK, як і раніше, працює на всіх платформах і є безкоштовною;

– Unity Ads - велика платформа для просування ігор і мобільних додатків. Охоплення - 1 млрд користувачів по всьому світу. Дозволяє запускати виключно міжсторінкових відеорекламу. Ролики інтегруються в продукти партнерів;

– Unity Multiplayer – дозволяє легко створювати додатки для комфортної гри в мультиплеєр. Платформа є досить новою і інтеграція з власним додатком відбувається дуже простим способом – натиснути одну кнопку для отримання результату і підключених скриптів, з якими можна уже в майбутньому створювати свій мультиплеєрний проект;

– Untiy Collaborate спрощує збереження, обмін і синхронізацію, забезпечуючи доступ до проектів Unity для всіх співробітників незалежно від їх місця розташування. Він підтримує хмарні сервіси і вбудований безпосередньо в Unity.

Екосистема Unity є досить багатою та допомагає розробникам у створенні власних додатків.

Отже, доповнена реальність є результатом використання технології для накладання інформації - звуків, зображень та тексту - на світ, який ми бачимо. Розглянуто програмні засоби за допомогою яких можна розробляти програмне забезпечення та було визначено, що для даної теми магістерської роботи добре використовувати Unity, було зроблено порівняльну характеристику засобі для зберігання файлів для майбутнього застосування, після чого було обрано Firebase, було проведено порівняння аналогів та порівняльна характеристика аналогів до нової розробки, описані плюси нової розробки.

2 РОЗРОБКА МЕТОДУ МІНІМІЗАЦІЇ ТОЧОК СКАНУВАННЯ

2.1. Схема методу сканування точок

Використання доповненої реальності широко використовується у великих та малих компаніях, які спеціалізуються у сфері архітектури, дизайну, доповнена реальність також знайшла місце у методах навчання дітей та у медичній галузі. Сканування точок для взаємодії з об’єктами, на приклад, для встановлення певного тексту на картинку – важлива складова додатку.

Загальний принцип роботи методу сканування точок відбувається за допомогою обробки вхідних даних камери, тобто картинки, яка додає до картинки чорно-білий фільтр та визначає певні місця, до яких може ухопитись точка.

Результатом такого процесу є велика кількість точок, які оброблюються процесором телефону, що не завжди дає належну швидкодію додатку.

Опис послідовності, що показує роботу даного методу наступний: на вхід подається картинка, до якої кріпиться фільтр Канні, який знаходить контури на рисунку, після цього її потрібно згладити, що відбувається за допомогою метода Гаусса, виконати полігональне приближення для знаходження точок, які можуть потім у майбутньому використатись для розставлення об’єктів.

Після даних процедур буде виконувати метод по мінімізації та обрахування точок задля побудови матриці, яка буде далі відображатись у додатку телефона як область, на якій можна відобразити об’єкти завантаженні з бази даних.

Схематичне представлення роботи методу сканування точок для побудови області відображено на рисунку 2.1.

Рисунок 2.1 – Схематичне представлення роботи методу пошуку точок

Перелік процесів, що показують метод сканування точок наведений нижче:

– процес отримання картинки з камери;
– накладання фільтру Канні, визначення контурів;
– зглажування картинки за допомогою метода Гаусса;
– знаходження точок для закріплення;
– мінімізація точок.

2.2. Процес сканування точок

Предметом роботи даної програми є методи та програмні засоби обчислення точок та відображення їх у додатку задля розміщення на них об’єктів та мінімізація їх. Запропонований метод дозволяє швидше обробляти область у якій дозволяється розмістити об’єкти.

Розглянутий вище алгортим роботи методу обрробки точок дозволяє побудувати область значно швидше, задля того щоб була можлвість розмістити об’єкти на даній області.

Процес будування області зводиться до наступних кроків:

– отримання всіх можливих точок;
– побудова області з мінімізованих точок.

Схематичне представлення роботи процесу побудови області відображене на рисунку 2.2.

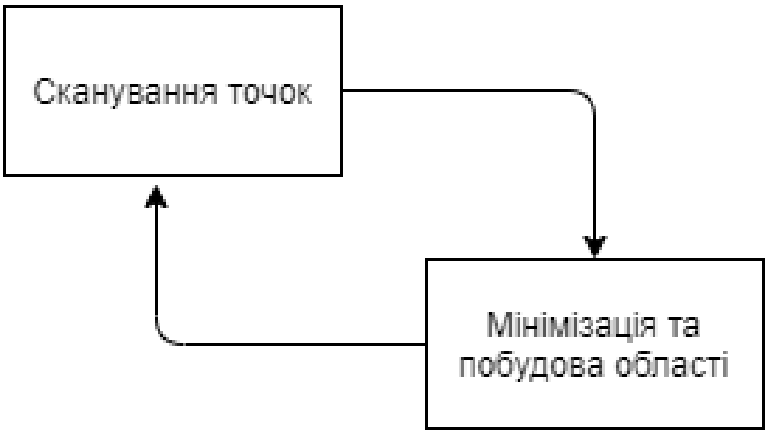


Рисунок 2.2 – Схематичне представлення роботи процесу побудови області

Перевагою даного методу є пришвидшення роботи сканування області за допомогою меншої кількості точок в порівнянні в отриманаю кількість точок після фільтрування та обробки зображення катринки отриманою з камери телефону.

Після модифікаї загальної схеми знаходження точок, що зображена на рисунку 3.1, виходить нова доповнена схема представлення роботи процесу сканування точок та будування області, що зображена на рисунку 2.3.

Рисунок 2.3 – Схематичне доповнене представлення роботи методу сканування точок та побудова області

2.3. Підрахунок точок та оптимізація за допомогою їх мінізації

Для того щоб встановити необхідні для додатку ресурси потрібно

визначити методи та ресурси для роботи процесу обробки точок, що були наведені в минулому підрозділі, з метою продемонструвати його реалізацію механізмами та компонентами за допомогою мови програмування C# та Unity3D. Після буде розглянуто проектування та створення користувацького інтерфейсу.

У підходах до відстеження на основі бачення особливостями зображення, що розглядаються для обчислення пози, часто є точки, лінії, контури або комбінація цих різних особливості. Для ілюстрації формалізму проблеми оцінки пози ми розглянемо випадок точкових ознак.

Нехай pi = (xi, yi, zi), i = 1,. . . , n, n ≥ 3 - це набір 3-D неколінеарних опорних точок, визначених у світовій системі відліку. Відповідні координати простору камер qi = (xi, yi, zi) задаються: qi = Rpi + T. R і T описують жорстке перетворення корпусу від світової системи координат до системи координат камери і саме такі параметри, пов'язані з камерою, створюють проблему.

Функція сканування точок, V є відображенням, на вхід якої подається оброблена на відфільтрована картинка, назвемо її I, а на виході ми отримуємо, відповідно, масив точок mt.

$$mt=V(I) \tag{1}$$

Код тоді складається з 16 біт і дозволяє 216 = 65536 можливих різних точок закріплення, що є важким процесом обробки для процессора телефону. Ціль поставленої задачі у мінімізація та відбиранні конкретних точок задля швидкої роботи додатку та сканування області.

Для мінімізації кількості точок використовується аналітичний алгоритм Дід’єр, його складність та час виконання невеликий та говорить він про правильну квадратну форму, що має властивість:

$$AB = CD \tag{2}$$

Після вхідних даних матриці точок маємо можливість скористатись алгоритмом та на виході отримаємо меншу кількість точок за допомогою обробки матриці поточних точок камери, формула виглядає наступним чином:

$$\begin{pmatrix} Ub-Uc & Ud \\ Vb-Vc & Vd \\ -1 & 1 \end{pmatrix} \begin{pmatrix} Zb \\ Zc \\ Zd \end{pmatrix} = \begin{pmatrix} Va \\ Ua \\ -1 \end{pmatrix}, \quad (3)$$

В результаті отриманої мінімізованої матриці точок є можливість побудувати нову область для можливості розмістити об’єкти. Блок схема роботи алгоритму та мінімізування точок зображено на блок схемі 2.4.

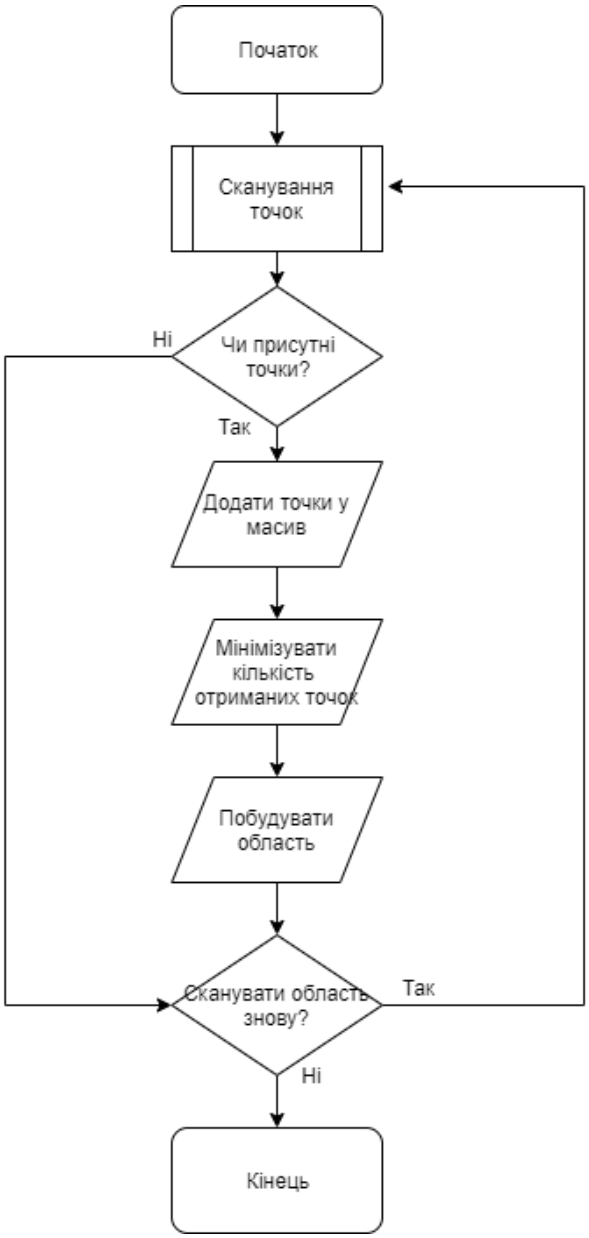


Рисунок 2.4 – Блок схема роботи алгоритму

2.4. Підрахунок пришвидшення процесу сканування точок

Підрахунок пришвидшення роботи алгоритмів буде проведено за допомогою порівняння алгоритму нової розробки та алгоритму програмного забезпечення IKEA Place.

Вирахування пришвидшення процесу буде відбуватись за формулою:

$$P = t * k_T, \quad (4)$$

де P – час, за який побудується область з усіх отриманих точок, t – стала, яка вказує на середній час обрахунку області по кожній точці та дорівнює 0.3 мілісекунд (мс), k_T – кількість отриманих точок.

Алгортим у IKEA Place, при скануванні області, повертає кількість точок k_T приблизно трьох тисяч точок на одну із областей для побудування.

Тому обрахунок часу обробки точок для побудови області за формулою дорівнюватиме:

$$P = 0{,}3 * 3000 = 900 \text{ мс}$$

Алгоритм нової розробки при скануванні області повертає кількість точок k_T приблизно в одну тисячу вісімсот точок на одну із областей, тому обрахунок часу обробки точок для побудови області за формулою дорівнюватиме:

$$P = 0{,}3 * 1800 = 540 \text{ мс}$$

Отже, нова розробка пришвидшує сканування та побудову області у 1.66 разів.

Таким чином, у даному розділі запропоновано метод і розроблено швидкодіючі алгоритми сканування поверхні, наведено формули для сканування зі зменшеною кількістю точок, розроблено блок схеми та послідовність реалізації алгоритмів, проведено розрахунки для отримання мінімізованої кількості точок, які показали, що коефіцієнт пришвидшення побудови точок у новій розробці становить 1.66.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Розробка та опис програми реалізації алгоритмів розв'язання задачі

Андроїд додаток буде працювати обробляючи введені дані акселерометра для можливості розвернути гравця на 360 градусів та визначення кута екрану телефона поворотами вліво та вправо[12]. Основною метою створення проекту є надання користувачу надати порожній кімнаті вигляду або додати щось у вже існуючу.
Збереження даних для подальшого їх опрацювання буде зберігатись у базі даних Firebase, що забезпечує можливість завантажити об'єкти. Для подальшого опрацювання цих об'єктів використовується Asset Bundle.

У грі буде забезпечена можливість повороту гравця, що виконується за допомогою датчиків, що знаходяться в телефоні. Акселерометр буде передавати координати положення телефону у просторі, гіроскоп буде передавати кут повороту телефону.

Проект використовуватиме доповнення, яке міститиме бібліотеки та класи з роботою з Google ARCore для Unity3D. Google ARCore дозволить кожному з розробників отримати задоволення від розробки ігор під телефони простим, веселим та доступним шляхом.

Для того, щоб у додатку використовувались інструменти доповненої реальності для телефонів потрібно у проекті вказати XR налаштування та поставити галочку на опції «ARCore Supported». Дане налаштування зображене на рисунку 3.1.

Рисунок 3.1 – Початкове налаштування для додатку у Unity

Додатково потрібно реалізувати анімацію переходу між предметами та елементами в яких потрібно змінювати матеріали. На блок схемі, що зображена на рисунку 3.2, зображена послідовність який дозволяє

користувачу обрати одну із доступних фурнітур та застосувати його у додатку.

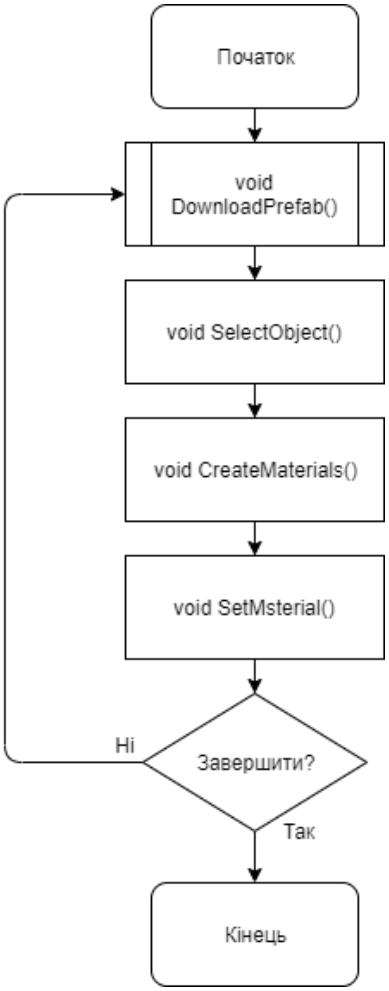


Рисунок 3.2 – Блок-схема створення об’єктів

Для кращого розуміння, що користувач буде робити у додатку після того, як його запустить, представлена Uml-діаграма, а саме діаграма прецедентів, що представлена на рисунку 3.3.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система

представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (англ. use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

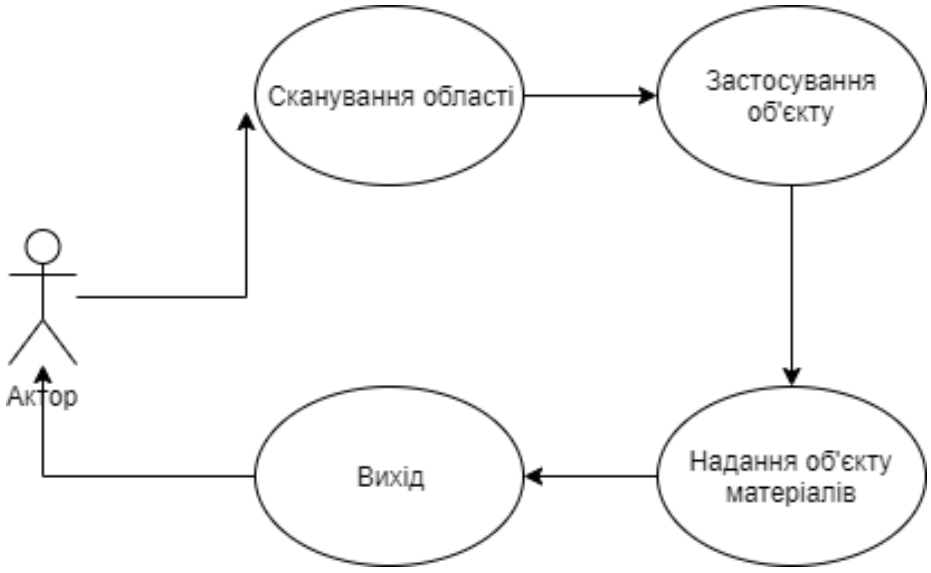


Рисунок 3.3 – Діаграма прецедентів

У цьому проекті було використано одну з композицій ООП – агрегація.

Агрегація – це створення об’єктів на існуючих класах як елементів інших класів. Про агрегація також часто говорять як про «відношення приналежності» за принципом «у машини є корпус, колеса і двигун».

Зазвичай вкладені об’єкти класу оголошуються закритими, що робить їх недоступними для прикладних програмістів, але творець класу може їх змінювати.

Одним з найголовніших класів є MonoBehaviour. Сам клас MonoBehaviour включає в себе всі функції для реалізації взаємодії між об’єктами. Від нього наслідуються такі класи як, UIManager, Anchor, DownloadHelper та інші. Вигляд діаграми класів представлений на рисунку

3.4.

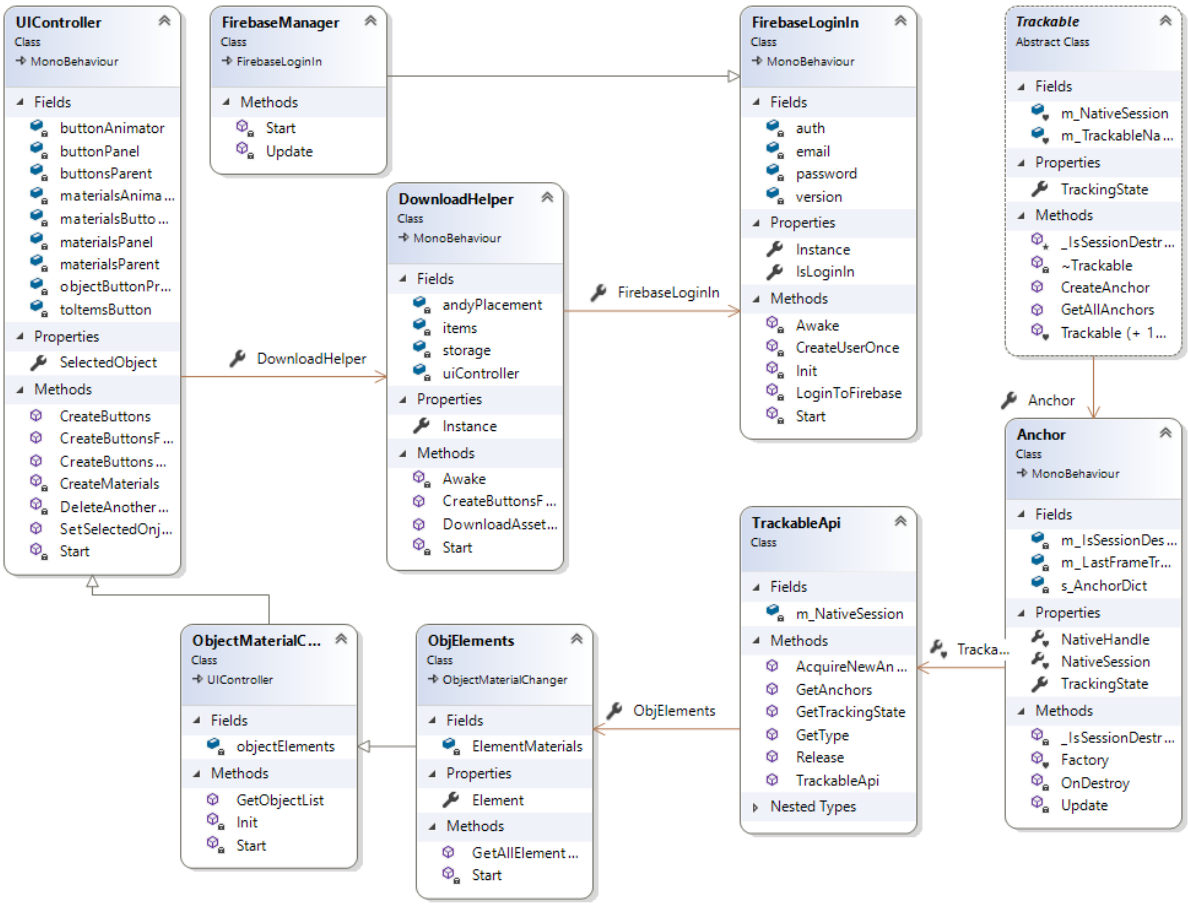


Рисунок 3.4 – Діаграма класів

Діаграми станів є добре відомим засобом опису поведінки систем. Вони визначають всі можливі стани, в яких може перебувати конкретний об'єкт, а також процес зміни станів об'єкта в результаті впливу деяких подій. Діаграма станів зображена на рисунку 3.5.

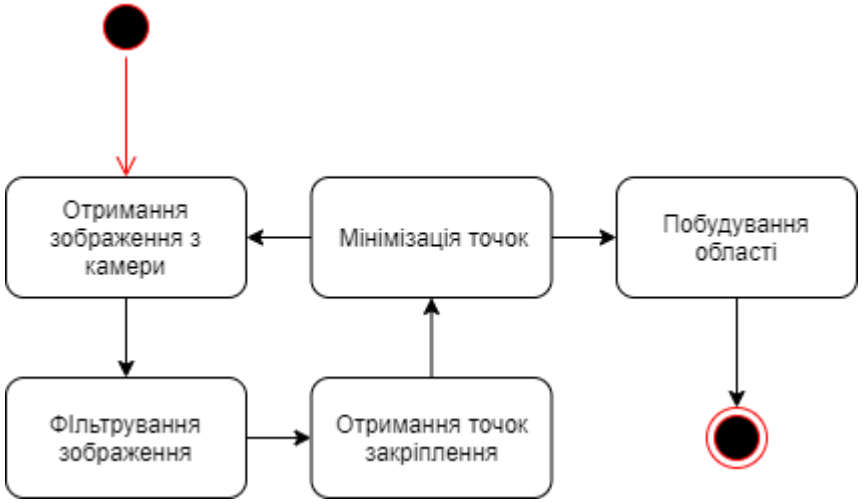


Рисунок 3.5 – Діаграма станів

Процес починається з початкової точки входу, потім слідує до стану «Отримання зображення з камери», після переходить до стану «Фільтрування зображення», що застосовує до зображення фільтри для отримання точок для можливості закріплення, після чого переходить у стан «Отримання точок закріплення», що дозволяє додатку зберегти точки для майбутніх дій. Далі відбувається перехід у стан «Мінімізація точок» що дозволяє зменшити кількість точок для наступного стану «Побудування області», що дозволяє отримати область для майбутніх маніпуляцій, а сама розташування завантажених об’єктів на дану область. Зі стану «Мінімізація точок» можливий перехід у початковий стан «Отримання зображення з камери» задля повторного циклу для будування більшої області.
Діаграми послідовностей використовуються для уточнення діаграм прецедентів, більш детального опису логіки сценаріїв використання. Це відмінний засіб документування проекту з точки зору сценаріїв використання. Діаграми послідовностей зазвичай містять об'єкти, які взаємодіють в рамках сценарію, повідомлення, якими вони обмінюються, і які повертаються результати, пов'язані з повідомленнями. Втім, часто повертаються результати позначають лише в тому випадку, якщо це не очевидно з контексту. Об'єкти позначаються прямокутниками з підкресленими іменами (щоб відрізнити їх від класів). Повідомлення (виклики методів) - лініями зі стрілками. Повертаються результати - пунктирними лініями зі стрілками. Прямокутники на вертикальних лініях під кожним з об'єктів показують "час життя" (фокус) об'єктів. Втім, досить часто їх не зображують на діаграмі, все це залежить від індивідуального стилю проектування. Діаграма послідовностей зображена на рисунку 3.6.

Рисунок 3.6 – Зображення діаграми послідовностей

Послідовність починається з входу у додаток, де користувач зможе почати процес сканування області, також він зможе покинути додаток, якщо він на даний момент не потрібен. Далі, за допомогою руху телефону, відбувається

процес побудови області за допомогою вище описаних методів, після чого у користувача з'являється можливість розмістити об'єкт на сцені, тобто на побудованій області. Після натиснення кнопки користувачу дається можливість розмістити об'єкт та здійснювати майбутні маніпуляції на ними.

3.2. Створення користувацького інтерфейсу

Для створення користувацького інтерфейсу об'єкт Canvas в Unity3D.

Canvas – це область, всередині якої розміщуються всі елементі користувацького інтерфейса. Тобто, коли створюється новий ігровий об'єкт, то Canvas для нього створюється автоматично[4].

Для його створення потрібно зайти у меню «Create» -> «UI» -> «Canvas», що зображене на рисунку 3.7.

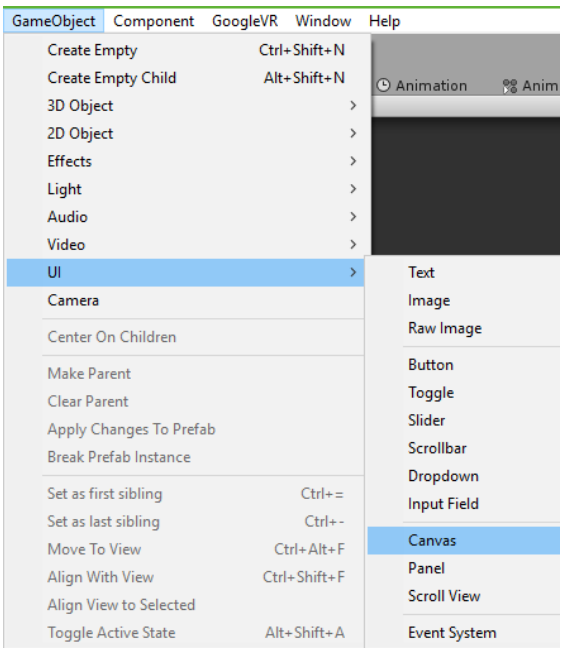


Рисунок 3.7 – Меню «UI» та пункт «Canvas»

В грі присутні три полотна, одне відповідає за основний вигляд, в якому міститься кнопки предметів, які доступні для завантаження об'єктів з бази даних.

Початковий канвас має вигляд, що зображений на рисунку 3.8, та містить інформацію про доступні об'єкти.

Рисунок 3.8 – Вигляд початкової панелі

Інше полотно відповідає за надання об’єкту вигляду за допомогою матеріалів до кожного з елементу об’єкта, що зображений на рисунку 3.9.

Рисунок 3.9 – Вигляд додаткового Canvas гравця

Для взаємодії користувача з меню використовується скрипт UIManager, в якому описана наступні функції:

– додання об’єктів до полотна для створення кнопок;

– програвання анімації для відображення чи приховання панелей;

– додання кнопок для відображення частин об’єкту;

– додання кнопок для відображення матеріалів кожної з частин об’єкту;

– видалення існуючих кнопок.

На ієрархії сцени він вкладений як компонент об’єкт головного канвасу. Для зручної взаємодії було зроблене полотно у нижній частині екрану задля уникнення незручностей для людей з невеликими руками при великій діагоналі єкрану телефону. Функції класу представлені нижче:

```
    private void CreateMaterials(ObjElements obj)
    {
        var materials = obj.GetAllElementMaterials();
        foreach (var mat in materials)
        {
            var materialButton = Instantiate(objectButtonPrefab, materialsParent);
            materialButton.GetComponentInChildren<Text>().text = "";
            materialButton.GetComponentInChildren<Image>().sprite =
Sprite.Create((Texture2D)mat.mainTexture, new Rect(0.0f, 0.0f,
mat.mainTexture.width, mat.mainTexture.height), new Vector2(0.5f, 0.5f), 100.0f);
            materialButton.onClick.AddListener(() =>
            {
                Material currentMaterial = mat;
                obj.Element.material = currentMaterial;
                Debug.Log("Selected material: " + mat.name);
            });
        }
        materialsAnimator.SetTrigger("OpenUIElement");
    }
    private void DeleteAnotherObjects(Transform materialsParent)
    {
        foreach (Transform child in materialsParent)
        {
            Destroy(child.gameObject);
        }
    }
```

3.3. Тестування додатку

Тестування проекту відбувається за допомогою «Unity Test Runner»[18]. Це інструмент, який перевіряє код як у режимі редагування, так і в режимі відтворення на цільових платформах, таких як ПК, Android та iOS. Для того, щоб зайти в нього потрібно зайти в пункт меню «Window» -> «Test Runner». Його вигляд зображений на рисунку 3.10.

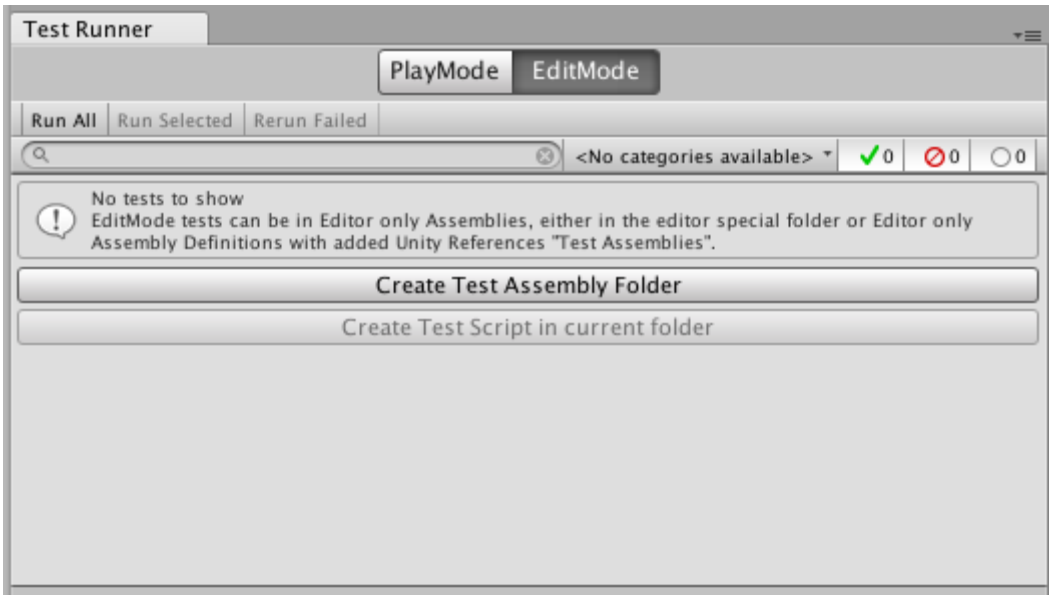


Рисунок 3.10 – Вигляд «Test Runner»

«Unity Test Runner» використовує інтеграцію Unity бібліотек NUnit бібліотеки, які є open-source модулем тестування для мов програмування на .Net.
UnityTestAttribute є основнм доповоненням до стандартної бібліотеки NUnit для Unity Test Runner. Це типовий тест, який дозволяє пропустити кадр з тесту. Використовувати UnityTestAttribute можна:

- в процесі гри;
- в режим редагування.

Якщо в проекті немає ніяких тестів, потрібно натиснути кнопку «Create Test Assmbly Folder» у поточній папці, щоб створити базовий тестовий скрипт. Ця кнопка залишається неактивною, якщо додавання тестового скрипту призведе до помилки компіляції. Налаштування тестового файлу зображений на рисунку 3.11.

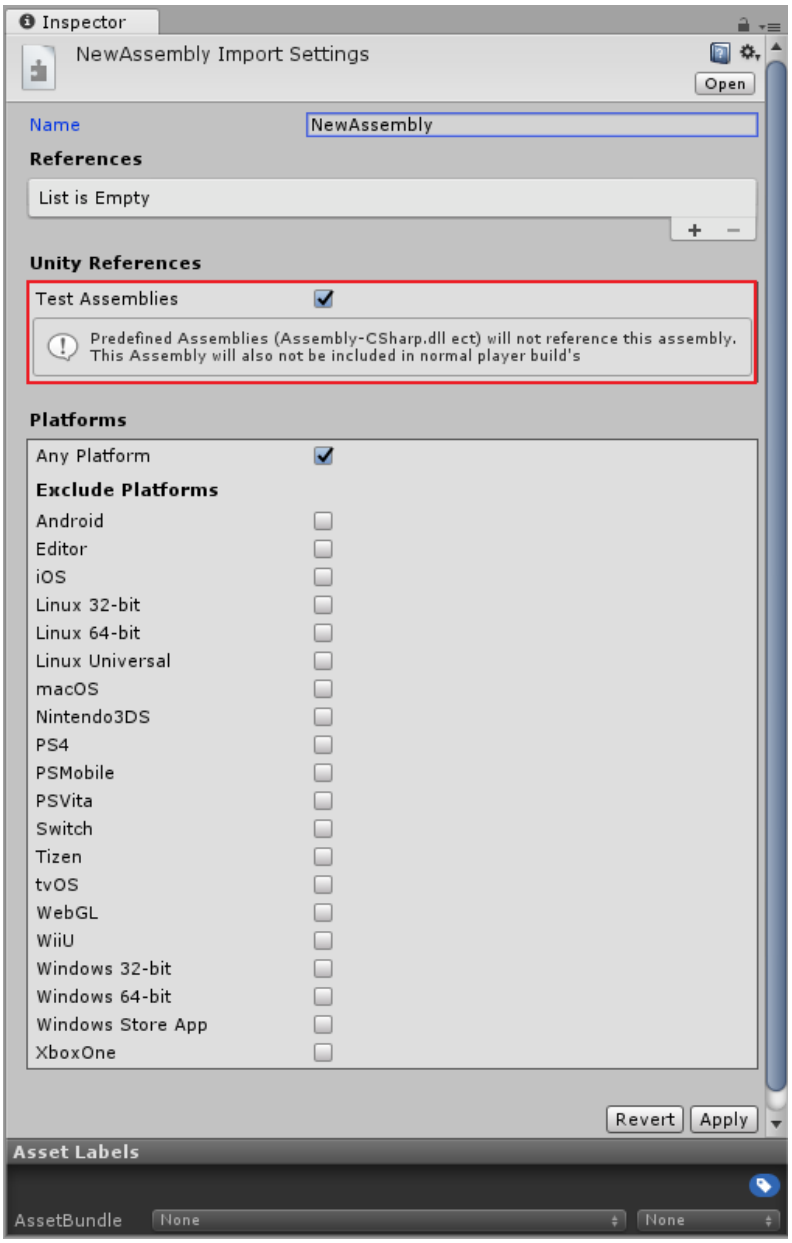


Рисунок 3.11 – Налаштування тестового файлу

Для прикладу потрібно створити тест, який показувати чи повертає функція позицію ворога чи ні. Для цього потрібно здійснити наступні дії:

– створити папку «Editor» у головній папці «Asset»;

– створити скрипт та надати йому ім’я;

У створену скрипті потрібно додати певний простір імен, який дозволятиме тестувати код чи функції «на льоту». Код наведений нижче:

```
using UnityEngine;
using NUnit.Framework;
```

Далі потрібно створити саму функцію, що виконуватиме тести з кодом прописаний програмістом. Код виглядає наступним чином:

```
[Test]
public void GetPointToEnemy_Test()
   {    }
```

Атрибут [Test] вказуватиме, що дана функція буде виконувати тільки в «Test Runner» та в процесі редагування коду. Якщо запустити функцію в «Test Runner», то він матиме вигляд зображений на рисунку 3.12.

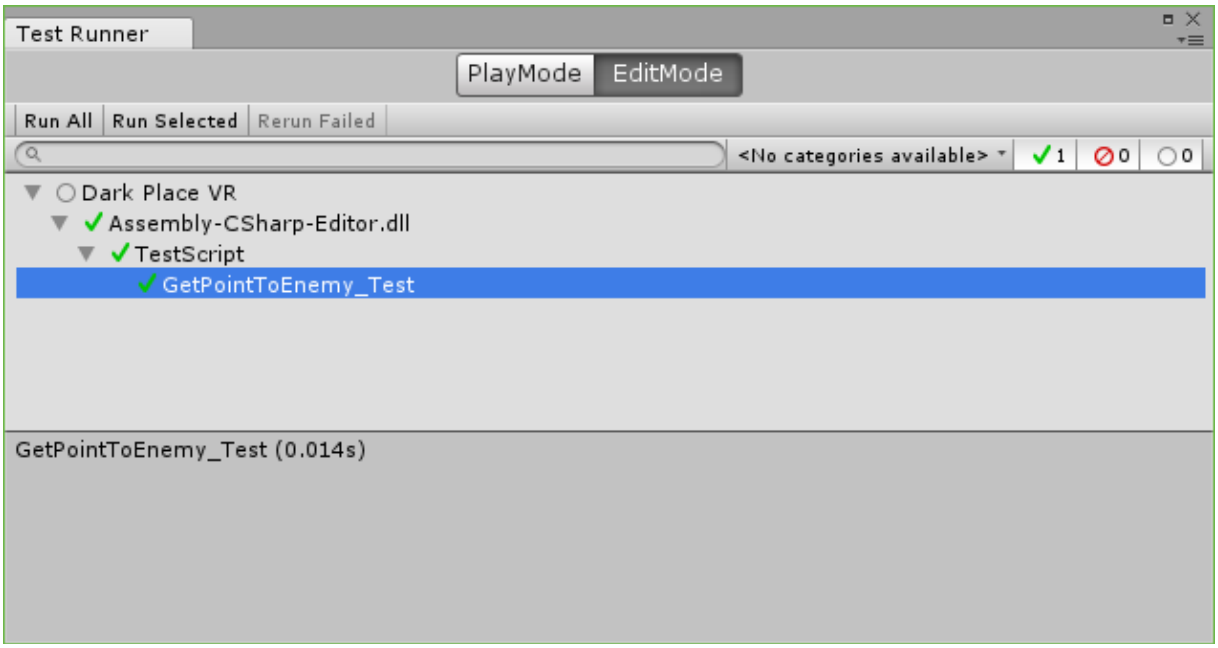


Рисунок 3.12 – Вигляд запущеного тестової функції

В результаті можна побачити, що функція виконалась за 0.014с, що є досить швидко та спрощує роботу програмістів для тестування їхнього власного коду.
Для перевірки коду отримання точки створення ворога, функція набуде наступного вигляду:

```
[Test]
   public void GetPointToEnemy_Test()
   {
      Vector3 center = Vector3.zero;
      float radius = 20f;

      float ang = Random.value * 360;
      Vector3 pos;
      pos.x = center.x + radius * Mathf.Cos(ang * Mathf.Deg2Rad);
      pos.y = center.y;
      pos.z = center.z + radius * Mathf.Sin(ang * Mathf.Deg2Rad);

      Assert.Pass("Fuction return {0} position of enemy", pos);
   }
```

Assert (твердження) є центральними елементами тестування в одному з фреймів xUnit, а NUnit - не є винятком. NUnit надає багатий набір тверджень як статичні методи класу Assert.
Якщо твердження не вдається, виклик методу не повертається і повідомляється про помилку. Якщо тест містить декілька тверджень, будь-

який, що слідує за помилкою, не буде виконаний. З цієї причини, як правило, краще всього спробувати одне твердження на тест.

Після завершення написання коду тестової функції, у «Test Runner» потрібно виділити певний тест, який потрібно виконати і натиснути «Run Selected». За малий проміжок часу функція буде виконана і «Test Runner» виведе повідомлення про помилку або про успішне завершення функції. На рисунку 3.13 зображено, що Unit-тест вивід повідомення про час, за який була виконана функція та твердження про те, що точка створення ворога була надана скрипту для подальших маніпуляцій на нею.

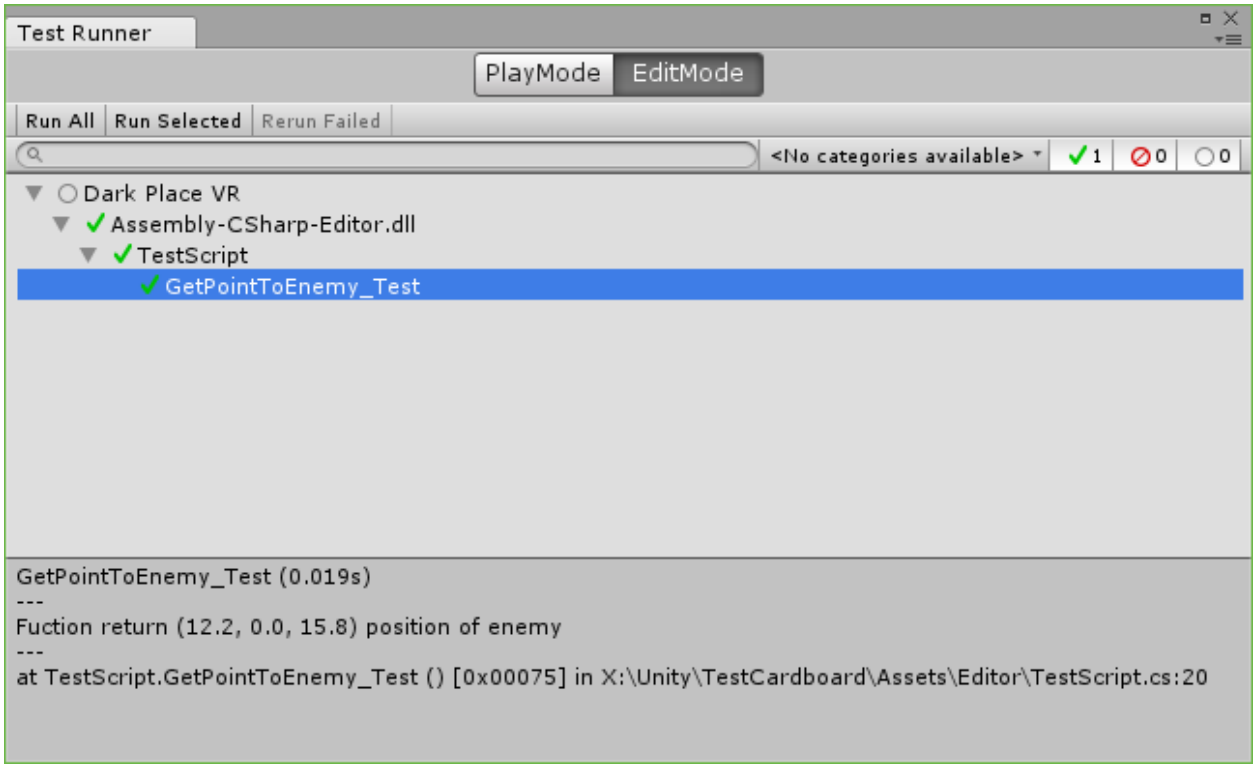


Рисунок 3.13 – Вигляд готової тестової функції

3.4. Класи та функції програми

На основі вже існуючих класів можна створити підкласи, які отримують (успадковують) функції та властивості батьківських класів. Кожна мова програмування може по своєму мати реалізацію даного методу об’єктно-орієнтованого програмування, проте у C# існує множинне успадкування, але не від класів, а успадкування від інтерфейсів. Множинне успадкування існує, скажімо, у мові програмування C++, коли наслідник може мати багато батьківських класів.

До складу програми реалізація коду взаємодії з графічним інтерфейсом, сканування області та взаємодія з об’єктами отриманих з бази даних.

Клас UIController – наслідуєься класом MonoBehaviour, виступає класом, який допомагає взаємодіяти користувачу з додатком, там має наступні властивості та методи.

Властивості:

– SelectedObject – об’єкт типу GameObject, який у майбутньому можна розмістити на згенеровану область;

– buttonsParent – об’єкт типу Transform, який є батьківським елементом для розміщення кнопок елементів об’єкту;

– objectButtonParent – об’єкт типу Button, це об’єкт, який має функціонал для взаємодії з об’єктом;

– materialsParent – об’єкт типу Transform, який є батьківським елементом для розміщення кнопок елементів об’єкту;

– materialsPanel та buttonPanel – панелі типу GameObject, які містимуть у собі батьківські елементи для розміщення кнопок;

– materialsAnimator, buttonAnimator – об’єкти типу Amimator, які у собі референси на Animation State Machine у Unity.

Функції:

– функція Start – стандартна функція у Unity, яка виконується при старті програми;

```
void Start()
  {
    materialsAnimator = materialsPanel.GetComponent<Animator>();
    buttonAnimator = buttonPanel.GetComponent<Animator>();
    toItemsButton.onClick.AddListener(() => {
DownloadHelper.Instance.CreateButtonsForItems(); });
  }
```

При виклику даної функції відбувається наступні дії:

1) materialsAnimator отримує компонент Animator з об’єкту materialsPanel, тим самим виконує ініціалізацію, об’єкт перестає бути null;
2) buttonAnimator отримує компонент Animator з об’єкту buttonPanel, тим самим виконує ініціалізацію, об’єкт перестає бути null;
3) Для toItemsButton задається «слухач», який отримує анонімну функцію, яка виконується при натисканні на дану кнопку, анонімна функція в свою чергу викликає функцію з «одинака» (Singleton) для створенна кнопок на головному екрані.

– SetSelectedObject – функція для встновлення SelectedObject;

```
    public void SetSelectedOnject(GameObject selectedObject)
        {
            SelectedObject = selectedObject;
            Debug.Log("Selected: " + SelectedObject.name);
        }
```

Дана функція має параметр selectedObject, який передається властивості SelectedObject, що у майбутньому задає головний об’єкт для маніпуляції над ним. В свою чергу, відбувається вивід інформації, який саме об’єкт був обраний для дій над ним.

– CreateButtons – функція, яка створює кнопки у батьківських елементах, також викликає анімацію даних об’єктів;

```
public void CreateButtons()
    {
        if (SelectedObject == null)
        {
            Debug.LogError("Selected object is null, please set gameObject to this field");
            return;
        }

        buttonAnimator.SetTrigger("CloseUIElement");
        materialsAnimator.SetTrigger("CloseUIElement");
        var objects = SelectedObject.GetComponent<ObjectMaterialChanger>().GetObjectList();
        Debug.Log(objects.Count);
        DeleteAnotherObjects(buttonsParent);
        foreach (var obj in objects)
        {
            var objectButton = Instantiate(objectButtonPrefab, buttonsParent);
            objectButton.GetComponentInChildren<Text>().text = obj.name;
            objectButton.onClick.AddListener(() =>
            {
                DeleteAnotherObjects(materialsParent);
                CreateMaterials(obj);
                Debug.Log("Selected " + obj.name);
            });
        }
        buttonAnimator.SetTrigger("OpenUIElement");
        materialsAnimator.SetTrigger("OpenUIElement");
    }
```

Дана функція працює майже з усіма властивостями, які доступні у класі та виконує наступні дії:

1) на початку виклику функції відбувається перевірка, чи не є вибраний об’єкт пустим, якщо він дійсно пустий або не створений

– функція завершується, якщо ні – переходить до наступного кроку;

2) далі відбувається виклик анімації, функція викликає так звані «тригери», які спрацьовують один раз за виклик, та виконують задану їм анімацію, а саме анімацію на панелями з кнопками;
3) далі функція отримує у локальну зміну об’єкти з якими можна буде виконувати манпуляції;
4) за допомогою циклу foreach, що є кращим з усіх так як виконує ту кількість ітерацій скільки об’єктів в переданому йому масиві об’єктів, та створює з префабу objectButtonPrefab клон кнопки, задає йому батьківський об’єкт, до якого він «чіпляється», задається ім’я кнопці згідно назві об’єкту та задає анонімну функцію, яка в свою чергу, при натисканні на кнопку, буде виконувати дії видалення попередніх кнопок, створить нові кнопки з матеріалами, які можна у майбутньому задати, та виводить у консоль, який саме об’єкт був вибраний.
5) далі виконується анімація, яка скриває панелі, та з’являється уже з новими кнопками.

– CreateButtonsForItems – функція для створення кнопок об’єктів, які розміщені в базі;

Дана функція схожа до реалізації з вище вказаним кодом, але логіка дещо інша, код наведений нижче.

```
public void CreateButtonsForItems(string[] items)
    {
        buttonAnimator.SetTrigger("CloseUIElement");
        materialsAnimator.SetTrigger("CloseUIElement");
        DeleteAnotherObjects(buttonsParent);
        DeleteAnotherObjects(materialsParent);
        foreach (var item in items)
        {
```

```
                var itemButton = Instantiate(objectButtonPrefab,
    buttonsParent);
                itemButton.GetComponentInChildren<Text>().text = item;
                itemButton.onClick.AddListener(() =>
                {

    StartCoroutine(DownloadHelper.Instance.DownloadAssetBundle(string.Em
    pty, item));
                });
            }
            buttonAnimator.SetTrigger("OpenUIElement");
        }
```

Дана функція виконує дію створення кнопок зі списку предметів, які знаходяться у базі даних Firebase. Послідовність наступна: спочатку виконується анімація об’єктів, потім через цикл foreach, який був описаний вище створює ту кількість кнопок, скільки предметів в базі даних, на кожну з кнопок задається анонімна фукнція, яка в свою чергу викликає карутіну, що виконує завантаження об’єкту з бази даних.

– DeleteAnotherObjects – функція для видалення кнопок в батьківських елементах;

```
private void DeleteAnotherObjects(Transform materialsParent)
  {
    foreach (Transform child in materialsParent)
    {
      Destroy(child.gameObject);
    }
  }
```

Дана функція видаляє усі дочірні об’єкти (кнопки), які знаходяться в батьківських панелях.

– CreateMaterials – функція для створення кнопок, за допомогою яких можна надати об’єктам вигляду.

```
private void CreateMaterials(ObjElements obj)
  {
    var materials = obj.GetAllElementMaterials();
    foreach (var mat in materials)
    {
      var materialButton = Instantiate(objectButtonPrefab, materialsParent);
      materialButton.GetComponentInChildren<Text>().text = "";
```

```
            materialButton.GetComponentInChildren<Image>().sprite =
Sprite.Create((Texture2D)mat.mainTexture, new Rect(0.0f, 0.0f,
mat.mainTexture.width, mat.mainTexture.height), new Vector2(0.5f, 0.5f), 100.0f);
            materialButton.onClick.AddListener(() =>
            {
                Material currentMaterial = mat;
                obj.Element.material = currentMaterial;
                Debug.Log("Selected material: " + mat.name);
            });
        }
        materialsAnimator.SetTrigger("OpenUIElement");
    }
```

Дана функція виконує створення кнопок для взаємодії з матеріалами на елементах об’єкту. Реалізація схожа на функцію CreateButtons.

Клас DownloadHelper виступає класом для взаємодії з об’єктами, а саме для завнтаження їх з бази даних.

Властивості:

– статична зміна DownloadHelper виступає у ролі посилання самого на себе, одним словом Singleton;

– storage – зміна типу FirebaseStorage, для посилання на базу даних;

– andyPlacement – зміна типу PawnManipulator, яка відповідає за надання класу вибраного об’єкту, та викнонує маніпуляції над об’єктом;

– uiController – зміна яка посилається на клас UIController.

Функції:

– функція Awake – стандартна функція Unity, виконується при запуску програми;

```
void Awake()
    {
        if (Instance == null)
        {
            Instance = this;
        }
        else if (Instance == this)
        {
            Destroy(gameObject);
        }
        DontDestroyOnLoad(gameObject);
        storage = Firebase.Storage.FirebaseStorage.DefaultInstance;
    }
```

Дана функція виконує ініціалізацію «одинака», до якого потім можна буде звернути через інстенс даного об’єкту, послідовність дій наступна: спочатку перевіряється чи існує інстенс, якщо ні – задаємо інстенсу посилання на

скрипт, якщо інстенс інсує і є тим самим скриптом – видаляємо інший об’єкт, потім викликаємо функцію DontDestroyOnLoad, яка вказує, що даний об’єкт не потрібно видаляти на сцені при переході чи інших маніпуляціях.

– Start – стандартна функція, виконується при старті програми, та виконує ініціалізацю об’єктів, шукає на сцені об’єкти типу UIController та PawnManipulator, код представлений нижче;

```
void Start()
{
    //CreateUserOnce();
    uiController = FindObjectOfType<UIController>();
    andyPlacement = FindObjectOfType<PawnManipulator>();
    CreateButtonsForItems();
    //StartCoroutine(DownloadAssetBundle("beds",
selectedPrefabToDownload));
}
```

– DownloadAssetBundle – функція для завантаження об’єкту з бази даних або, якщо об’єкт існує в кеші – витягує його звідти.

На початку виклику даної функція відбувається перевірка чи готовий кеш, код наведений нижче:

```
while (!Caching.ready)
    {
        yield return null;
    }
```

Після виконується перевірка чи здійснений вхід до Firebase для отрмання звідти об’єктів, код предствлений нижче:

```
while (!FirebaseLoginIn.Instance.IsLoginIn)
    {
        yield return null;
    }
```

Якщо ці дві умови повертаю правдиве значення, створюється зміна типу WWW, яка дозволяє завантажити об’єкт з бази даних, завантаження відбувається наступним чином:

```
WWW www = null;
    string url = string.Empty;
    storage = Firebase.Storage.FirebaseStorage.DefaultInstance;
    Firebase.Storage.StorageReference storageReference =
storage.GetReferenceFromUrl("gs://myarproject-f4b82.appspot.com/" +
prefabName);
    storageReference.GetDownloadUrlAsync().ContinueWith((Task<Uri> task)
=>
    {
        if (!task.IsFaulted && !task.IsCanceled)
        {
```

```
            Debug.Log("Download url: " + task.Result);
            url = task.Result.ToString();
        }
        else
        {
            Debug.Log(task.Result);
        }
    });
```

Функція звертається до інстенсу бази даних, задля отримання силки на об’єкт, який ми збираємось завантажити, якщо все добре – повертається посилання, якщо ні – виводиться помилка в консолі розробника.
Якщо немає ніяких помилок відбувається завантаження файлу та розпаковка його за допомогою assetBundle, даний код наведений нижче:

```
while (string.IsNullOrEmpty(url))
    {
        yield return null;
    }
    www = WWW.LoadFromCacheOrDownload(url, 0);
    yield return www;
    if (!string.IsNullOrEmpty(www.error))
    {
        Debug.LogError(www.error);
        yield break;
    }
    Debug.Log("FILE WAS DOWNLOAD!");
    var assetBundle = www.assetBundle;
    var gameObj = assetBundle.LoadAssetAsync(prefabName,
typeof(GameObject));
    yield return gameObj;
    andyPlacement.PawnPrefab = gameObj.asset as GameObject;
    assetBundle.Unload(false);
```

Клас ManipulationSystem – виступає в ролі системи для маніпуляції з об’єктами.
Властивості:

- Instance – властивість, яка повертає силку на самого себе, тобто на клас ManipulationSystem;
- DragGestureRecognizer – повертає посилання на екзмепляр класу DragGestureRecognizer;
- PinchGestureRecognizer – повертає посилання на екземпляр класу PinchGestureRecognizer;
- TwoFingerDragGestureRecognizer – повертає посилання на екземпляр класу TwoFingerDragGestureRecognizer;

– TapGestureRecognizer – повертає посилання на екземпляр класу TapGestureRecognizer;

– TwistGestureRecognizer повертає посилання на екземпляр класу TwistGestureRecognizer.

Функції:

– Update – стандартна функція в Unity, виконується після кожного фрейму та виконує апдейти маніпуляції з об’єктами;

```
public void Update()
{
   DragGestureRecognizer.Update();
   PinchGestureRecognizer.Update();
   TwoFingerDragGestureRecognizer.Update();
   TapGestureRecognizer.Update();
   TwistGestureRecognizer.Update();
}
```

Дана функція виконує наступні дії:

1) виконує дія, яка перевіряє кожен кадр чи користувач починає маніпуляція пересування об’єкту;
2) виконує дія, яка перевіряє кожен кадр чи користувач починає маніпуляція зміна розміру об’єкту;
3) виконує дія, яка перевіряє кожен кадр чи користувач починає маніпуляція підняття об’єкту;
4) виконує дія, яка перевіряє кожен кадр чи користувач починає маніпуляція виділення об’єкту;
5) виконує дія, яка перевіряє кожен кадр чи користувач починає маніпуляція обертання об’єкту.

– Deselect – функція для зняття активного об’єкту, з яким відбувались маніпуляції, наведена нижче;

```
internal void Deselect()
{
   SelectedObject = null;
}
```

– Select – функція для вибору об’єкту для здійснення майбутніх маніпуляцій, код наведений нижче.

```
internal void Select(GameObject target)
{
```

```
        if (SelectedObject == target)
        {
           return;
        }

        Deselect();
        SelectedObject = target;
        Debug.LogError("Selected");

uiController.SetSelectedOnject(target.GetComponentInChildren<ObjectMaterialC
hanger>().gameObject);
        StartCoroutine(uiController.CreateButtonsWithDelay(.1f));
    }
```

Дана функція виконує дію виділення об’єкту, об’єкт задається як головний. При цьому відбувається перевірка, чи об’єкт який потрібно виділити не є тим сама об’єкт, який вже виділений, якщо так – функція завершується, якщо ні – старий об’єкт перестає бути виділеним, виділяється новий об’єкт, після чого створюються кнопки для взаємодії з елементами головного об’єкту.

TracableApi – клас, який отримує та мінімізує точки отримані в результаті сканування області.

Властивості:

m_NativeSession – властивість, яка створює та повертає активну сесію для сканування області;

Функції:

– TrackableApi – конструктор для створення взаємодії з точками в активній сесії;

– GetTrackingState – повертає активний статус треку області;

– AcquireNewAnchor – створює точки до яких можна в майбутньому приєднати об’єкт;

– GetAnchors – повертає точки після мінімізації, код наведений нижче;

```
public void GetAnchors(IntPtr trackableHandle, List<Anchor> anchors)
    {
        IntPtr anchorListHandle = m_NativeSession.AnchorApi.CreateList();
        ExternApi.ArTrackable_getAnchors(
           m_NativeSession.SessionHandle, trackableHandle, anchorListHandle);

        anchors.Clear();
        int anchorCount =
m_NativeSession.AnchorApi.GetListSize(anchorListHandle);
        for (int i = 0; i < anchorCount; i++)
        {
```

```
            IntPtr anchorHandle =
               m_NativeSession.AnchorApi.AcquireListItem(anchorListHandle, i);
            Anchor anchor = Anchor.Factory(m_NativeSession, anchorHandle,
false);
            if (anchor == null)
            {
               Debug.LogFormat("Unable to find Anchor component for handle
{0}", anchorHandle);
            }
            else
            {
               anchors.Add(anchor);
            }
         }

         m_NativeSession.AnchorApi.DestroyList(anchorListHandle);
      }
```

Дана функція виконує пошук точок, мінімізує їх та додає у список, послідовність наступна: в активній сесії пошуку створюється масив, який буде використовуватись для зверігання точок, отриманих з картинки, арі, яке знаходиться у ядрі знаходить точки та повертає їх масив, після чого фабрика точок робить меншу кількість із вже інсуючих точок, щоб робить їх меншу кількість та додає до масиву.

Клас FirebaseLoginIn виступає в ролі помічника для здійсненян входу у базу даних Firebase.

Властивості:

– Instance – зміна типу FirebaseLoginIn, посилання на самого себе для швидкого доступу до функціоналу, так званий Singleton;

– IsLoginIn - зміна типу bool, яка вказує чи відбулась авторизація;

– Auth – зміна типу FirebaseAuth, яка допомагає здійснити вхід до бази даних;

– Email – зміна типу string, в якій зберігаєть стрічка з поштовою скринькою користувача;

– Password – зміна типу string, в якій зберігається зашифрований пароль для входу;

– Version – зміна типу int, яка вказує на версію об’єктів бази даних.

Функції:

– Awake – стандартна функцыя Unity, виконується при запуску програми;

```
   void Awake()
```

```
{
    if (Instance == null)
    {
        Instance = this;
    }
    else if (Instance == this)
    {
        Destroy(gameObject);
    }
    DontDestroyOnLoad(gameObject);

    Init();
}
```

Дана функція виконує ініціалізацію об’єкту, «одинака», до якого можна потім звернутись напряму без створення будь-яких посилань та екземплярів класу. Якщо «одинак» існує – видалити об’єкт, якщо «одинака» немає і він пустий – задати йому посилання на сам скрипт.

– CreateUserOnce – функція, яка дозволяє здійснювати вхід до бази даних, якщо користувача не існує – виконує дію створення нового користувача;

```
private void CreateUserOnce()
{
    auth.CreateUserWithEmailAndPasswordAsync(email,
password).ContinueWith(task =>
    {
        if (task.IsCanceled)
        {
            Debug.LogError("CreateUserWithEmailAndPasswordAsync was
canceled.");
            return;
        }
        if (task.IsFaulted)
        {
            Debug.LogError("CreateUserWithEmailAndPasswordAsync
encountered an error: " + task.Exception);
            return;
        }

        // Firebase user has been created.
        Firebase.Auth.FirebaseUser newUser = task.Result;
        Debug.LogFormat("Firebase user created successfully: {0} ({1})",
            newUser.DisplayName, newUser.UserId);
    });
}
```

Даний код (функція) виконує створення користувача, якщо при вході даного користувача не існує. Послідовність наступна: інстенс входу у базу даних виконує функцію створення користувача, якщо користувач створений – видає повідомлення, що новий користувач створився, якщо виникли якісь помилки при створенні – видає помилку в консолі.

– Init – функція для ініціалізації даних;

– LoginToFirebase – функція яка хдійснює вхід до бази даних;

```
private void LoginToFirebase()
{
    Firebase.Auth.Credential credential =
        Firebase.Auth.EmailAuthProvider.GetCredential(email, password);
    auth.SignInWithCredentialAsync(credential).ContinueWith(task =>
    {
        if (task.IsCanceled)
        {
            Debug.LogError("SignInWithCredentialAsync was canceled.");
            CreateUserOnce();
            LoginToFirebase();
            return;
        }
        if (task.IsFaulted)
        {
            Debug.LogError("SignInWithCredentialAsync encountered an error: " +
task.Exception);
            CreateUserOnce();
            LoginToFirebase();
            return;
        }

        Firebase.Auth.FirebaseUser newUser = task.Result;
        Debug.LogFormat("User signed in successfully: {0} ({1})",
            newUser.DisplayName, newUser.UserId);

        IsLoginIn = true;
    });
}
```

Дана функція виконує вхід до бази даних для дозволу користувачу завантажувати об’єкти з бази даних. Послідовність наступна: інстенс входу виконує функція входу користувача, якщо вхід дозволений – продовжує працювати та задає змінній IsLoginIn значення true, якщо ні – створює нового користувача, та виконує рекурсію, намагаєть увійти знову.

– Start – стандартна функція в Unity, яка виконується при старті програми.

Клас ObjectMaterialChanger – клас для здійснення кастомізації об’єктів;

Властивості: objectElement – масив об’єктів, з яких складається головний об’єкт.
Функції:

– GetObjectList – повертає масив об’єктів, з яких складається головний об’єкт, код наведений нижче;

```
public List<ObjElements> GetObjectList()
{
    return objectElements;
}
```

– Init – виконує ініціалізацію пошуку об’єктів;

– Start – стандартна функція в Unity, що виконується при старті програми.

Клас ObjElements – клас для дочірних елементів батьківського об’єкту, для маніпуляцій над ними.
Властивості:

– Element – зміна типу Renderer, використовується для надання об’єкту нового матеріалу , тощо;

– ElementsMaterials – масив об’єктів типу Material, використовується для зберігання можливих матеріалів, які можна застосувати для елементів об’єкту.

Функції:

– GetAllElementMaterials – функція, яка повертає матеріали для мабутнього застовування, функція представлена нижче;

```
public List<Material> GetAllElementMaterials()
{
    return ElementMaterials;
}
```

– Start – функція, яка викликається при старті програми.

У даному розділі розроблено мобільне програмне забезпечення для відображення предметів сканованої поверхні, реалізовано інтерфейс програми, проведено тестування додатку та описано структуру і деталі його реалізації, а саме – описано класи, значення, що використовуються у цих класах та методи, які оброблюють запропоновані алгоритми.

ВИСНОВКИ

В процесі дипломного проектування реалізовано поставлену задачу – проведено огляд предметної області розробки бізнес-додатків доповненої реальності з викроситання Google ARCore. Використана мова програмування C# та програмна платформа Unity3D. Набуто знання про створення додатків з використанням доповненої реальності. Також проаналізовано такі найбільш поширені шаблони проектування в Unity, як «Singleton» («Одинак») та «Object Pool» («об'єктний пул»).

Під час проектування додатку доповненої реальності для Android визначено задачі розробки та проаналізовано доцільність створення програмного забезпечення на основі платформи Unity. Вивчено всі переваги та недоліки даної платформи. Розглянуто особливості створення дизайну додатку за допомогою Canvas елементів у Unity та взаємодією між ними. За специфікацією розробки даного ПЗ основною мовою програмування було обрано мову C# і середовище розробки Unity та Visual Studio.

У другому розділі запропоновано метод і розроблено швидкодіючі алгоритми сканування поверхні, наведені формули для сканування та мінімізації точок, наведені блок схеми та послідовність реалізації алгоритму, проведені підрахунки точок та їх оптимізація, проведені підрахунки пришвидшення побудови точок нової розробки, які показали пришвидшення побудови області сканування у 1.66 разів.

У третьому розділі запропоновано метод і розроблено швидкодіючі алгоритми сканування поверхні, наведено формули для сканування зі зменшеною кількістю точок, розроблено блок схеми та послідовність реалізації алгоритмів, проведено розрахунки для отримання мінімізованої кількості точок, які показали, що коефіцієнт пришвидшення побудови точок у новій розробці становить 1.66.

Таким чином поставлена мета пришвидшення сканування області камерою реального світу була досягнута.

СПИСОК ДЖЕРЕЛ ПОСИЛАННЯ

1. **Android Design Patterns: Interaction Design Solutions for Developers / Greg Nudelman / Wiley; 1 edition (March 11, 2013) – 456 c.**
2. **Artificial Intelligence for Games / Ian Millington / CRC Press; 2 edition, 2015 – 896 c.**
3. **C# 7.0 in a Nutshell / Ben Albahari, Joseph Albahari / O'Reilly Media, 2017 – 1090 c.**
4. **CLR via C#. Програмування на платформі Microsoft.NET Framework 4.5 на мові C# / Джеффрі Ріхтер // Навчальний посібник – Санкт-Петербург: Питер, 2017. / ISBN 978-5-496-00433-6 – 896 с.**
5. **Designing Virtual Worlds / Richard Bartle / ISBN: 0-13-101816-7, 2014 – 575 c.**
6. **DirectX: продвинутая анимация / Джим Адамс / КУДИЦ-Образ, 2014 – 480 с.**
7. **Game Development Essentials: An Introduction / Jeannie Novak / Delmar Cengage Learning, 2012 – 512 c.**
8. **Game Programming Patterns / Robert Nystrom / Genever Benning; 1 edition (November 2, 2014) – 354 c.**
9. **Learning Unity Android Game Development / Thomas Finnegan / Packt Publishing - ebooks Account (April 30, 2015) – 346 c.**
10. **Learning Virtual Reality / Tony Parisi // Навчальний посібник – США, 2015. / O'Reilly Media, Inc (1005 Gravenstein Highway North, Sebastopol, CA 95472) – 172 c.**
11. **Mathematics for 3D Game Programming and Computer Graphics / Eric Lengyel / Course Technology PTR, 2012. - 545 c.**
12. **Mobile Game Development with Unity: Build Once, Deploy Anywhere / O'Reilly Media; 1 edition (September 4, 2017) – 464 c.**
13. **Professional Android 4 Application Development / Wrox; 3rd edition (May 1, 2012) – 864 c.**
14. **Unity Game Development Essentials / Will Goldstone / Packt Publishing, 2014 – 316 c.**
15. **Unity Game Development in 24 Hours / Mike Geig / Sams Publishing; 1 edition**

(December 6, 2013) – 400 с.

16. **Unity Virtual Reality Projects / Linowes J. // Публікація пакетів – США, 2016. / ISBN: 978-1-78398-855-6 – 286 с.**
17. **Искусственный интеллект в компьютерных играх. Как обучить виртуальные персонажи реагировать на внешние воздействия / Алекс Дж. Шампандар / Вильямс, 2014 – 768 с.**
18. **Мова програмування C# 6.0 та платформа .NET 4.6 / Ендрю Троелсен, Філіпп Джепікс / Expert's Voice – Вильямс, 2016 – 1440 с.**
19. **Программирование игр и головоломок / Ж. Арсак / «Наука, Главная редакция физико-математической литературы», 2013 – 222 с.**
20. **Разработка и отладка шейдеров / Алексей Боресков / БХВ-Петербург, 2015 – 488 с.**

ДОДАТКИ

Додаток А

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ

_____

_

«____» _____ 20

року

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської дипломної роботи

«МОБІЛЬНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ВІЗУАЛІЗАЦІЇ АРХІТЕКТУРНИХ ПРОЕКТІВ З ВИКОРИСТАННЯМ ДОПОВНЕНОЇ РЕАЛЬНОСТІ»

08-023.ДР.025.00.000.ТЗ

Виконав: студент 1 курсу, групи 2КІ-18м

зі спеціальності:
123 «Комп'ютерна інженерія»
(шифр і назва напряму підготовки)

Кокошко Станіслав Андрійович
(прізвище та ініціали)

Керівник: Черняк О.І.
(прізвище та ініціали)

м. Вінниця – 2020 р.

1. Підстава для виконання дипломної роботи (ДР)

1.1 Предметом роботи даної програми є методи та програмні засоби обчислення точок та відображення їх у додатку задля розміщення на них об’єктів.

1.2 Наказ про затвердження теми дипломної роботи.

2. Мета і призначення ДР

Підсумком виконання даної дипломної роботи стала програмна реалізація додатку для розміщення об’єктів на відсканованій області за допомогою доповненої реальності, що дозволяє пришвидшити роботу додатку та обрахунку точок для створення цієї області. З метою спростити роботу користувача був розроблений програмний продукт із зручним графічним інтерфейсом.

Високий рівень виконання поставленої задачі вирішено засобами мови програмування C#.

Основними перевагами є:

– розроблено новий метод сканування області;

– вдосконалена формула обчислення точок задля створення області у доповненій реальності;

– розроблений програмний додаток для сканування та розміщення об’єктів.

3. Вихідні дані для виконання ДР

Список технічної літератури, аналіз, вивчення та дослідження процесів сканування області за допомогою доповненої реальності, технічне завдання на магістерську роботу.

4. Матеріали, що подаються до захисту ДР

Пояснювальна записка ДР, графічні і ілюстративні матеріали, протокол попереднього захисту ДР на кафедрі, відгук наукового керівника, відзив рецензента, протоколи складання державних екзаменів, анотації до ДР українською та іноземною мовами.

5. Техніко-економічні показники

5.1 Витрати на програмні засоби, що використовуються в ході даної розробки, повинні бути мінімальними.

5.2 Лімітна ціна на програмне забезпечення не повинна перевищувати ціну аналога.

6. Порядок контролю виконання та захисту ДР

6.1.Робота виконується в три етапи, таблиця 6.1.

Таблиця 6.1 – Етапи виконання роботи.

Етап	Зміст	Початок	Кінець	Результат
1	Інформаційний пошук та огляд літературних джерел. Розробка методу та програмної реалізації блочного порівняння файлів.			Розділи 1 та 3.
2	Техніко-економічне обґрунтування теми дипломної роботи, розробка програмного засобу.			Чернетки матеріалів 1 розділу. Попередній захист.
3	Підготовка матеріалів пояснювальної записки.			Пояснювальна записка.

7. Загальний алгоритм роботи програми є таким:

1) Отримуємо зображення з камери.
2) Фільтруємо зображення за допомогою алгоритму Канні, для отримання границь.
3) Сглажуємо зображення за допомогою методу Гаусса.

4) Отримуємо точки з зображення.
5) Викликаємо функцію V.
6) Функція повертає мінімальну кількість точок.
7) Будує область.
8) При потребі та руху телефону переходимо на крок 1 задля отримання більшої області.
9) Розміщуємо об’єкти на дані області.

8. Порядок контролю та прийому

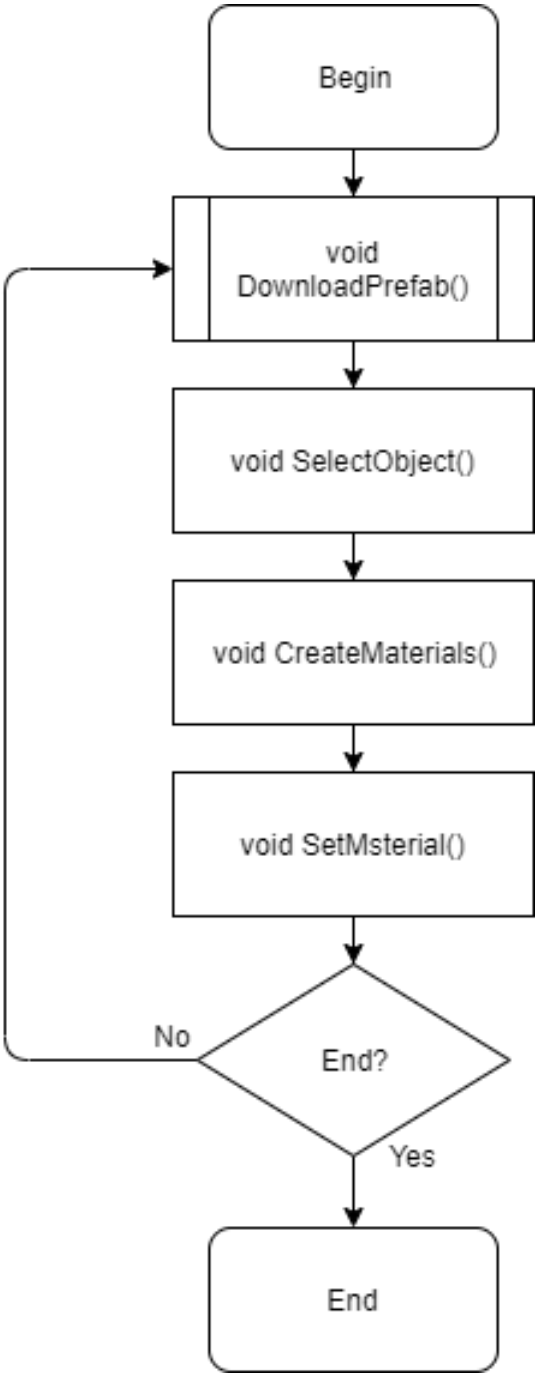
8.1 До приймання дипломної роботи надається:

- пояснювальна записка з відповідними узгодженнями;
- демонстрація програмного засобу;
- презентація;
- відгук керівника роботи та рецензента;

Технічне завдання до виконання прийняв _____ Кокошко С.А

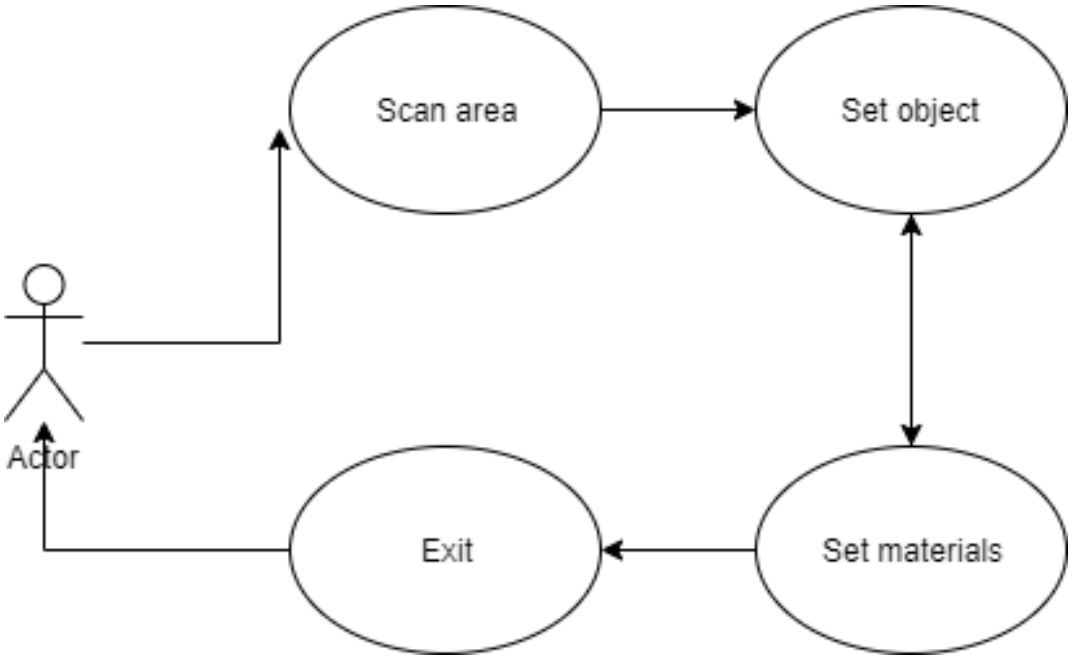
Додаток Б

Блок-схема створення об’єктів



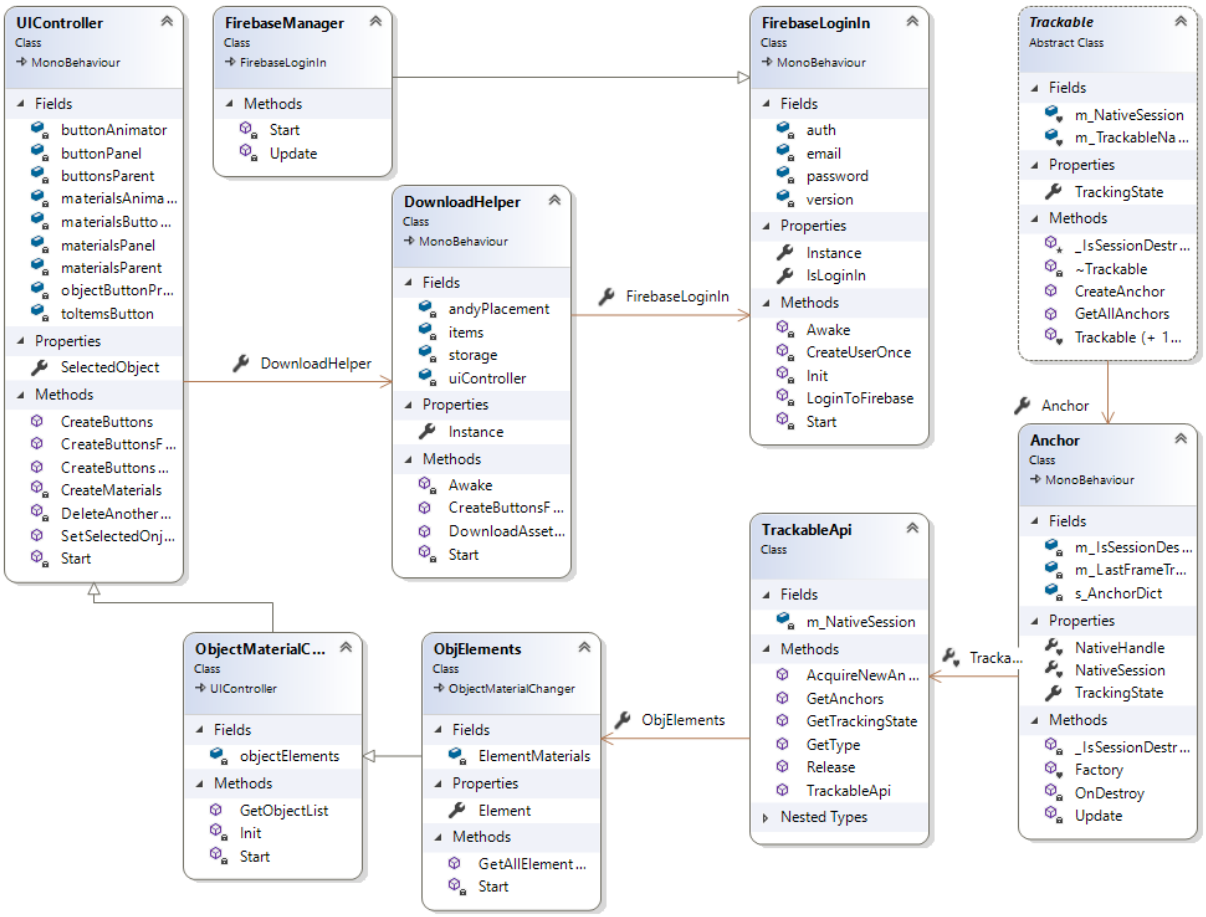
Додаток В

Діаграма прецедентів



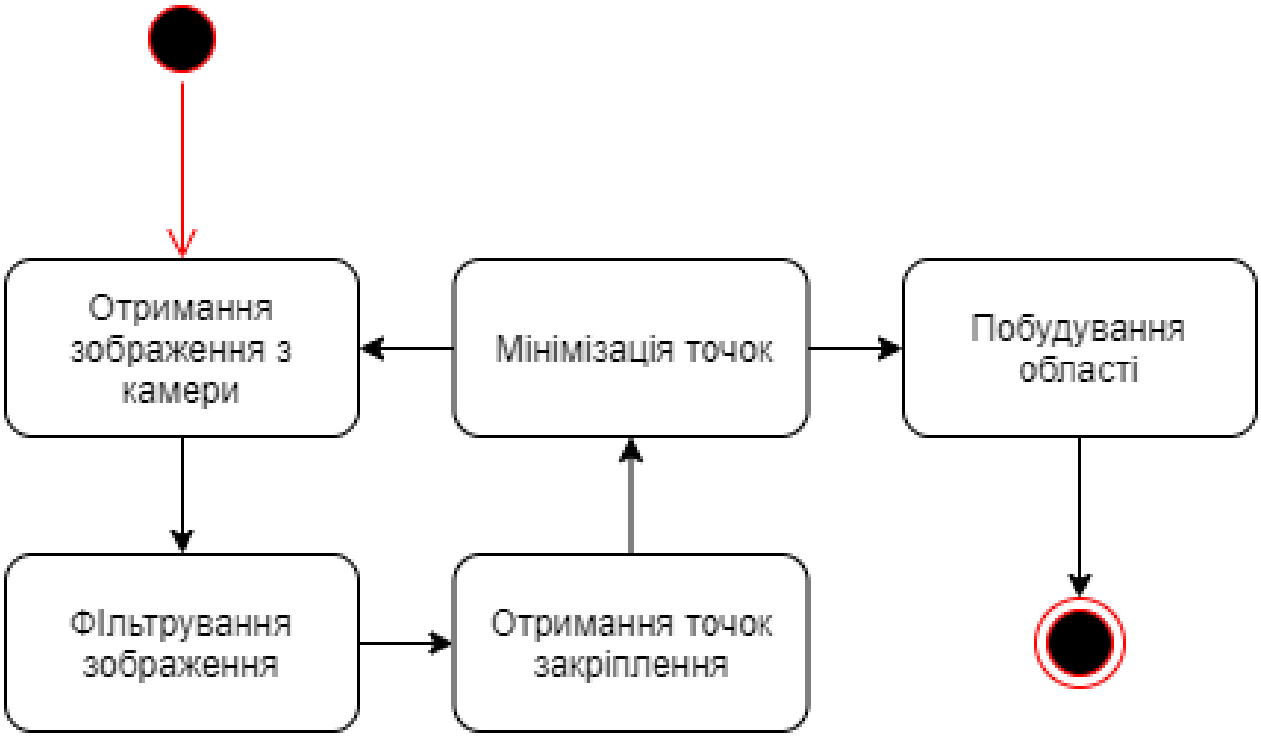
Додаток Г

Діаграма класів



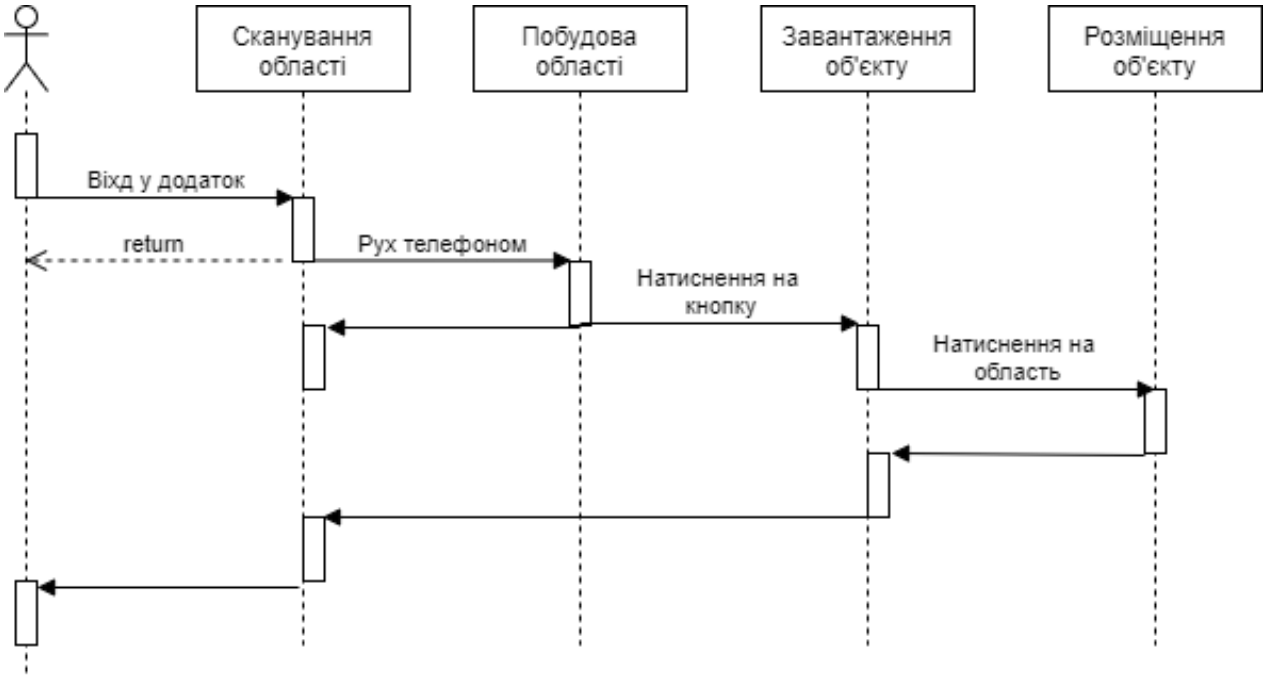
Додаток Д.

Діаграма станів



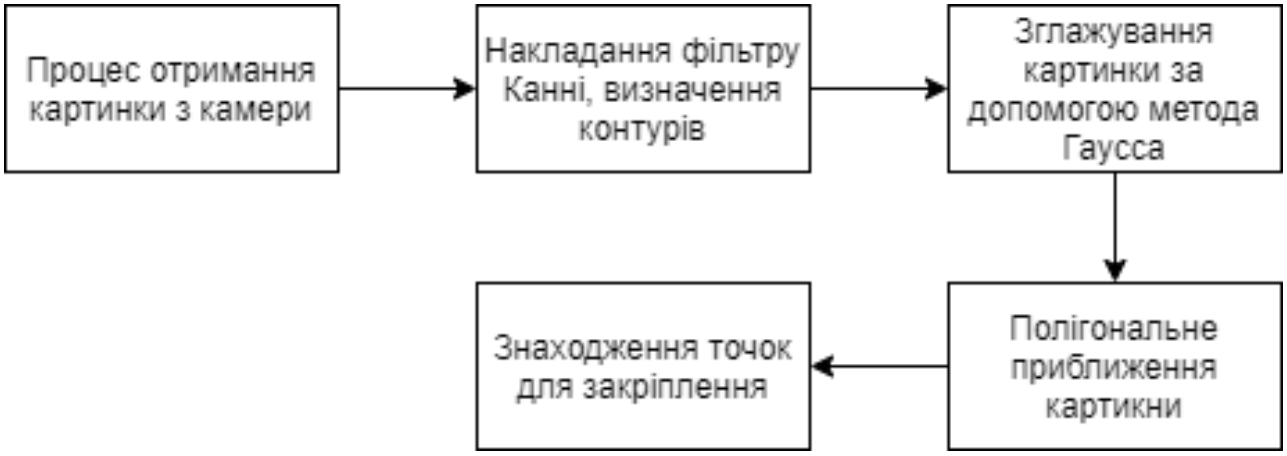
Додаток Е

Зображення діаграми послідовностей



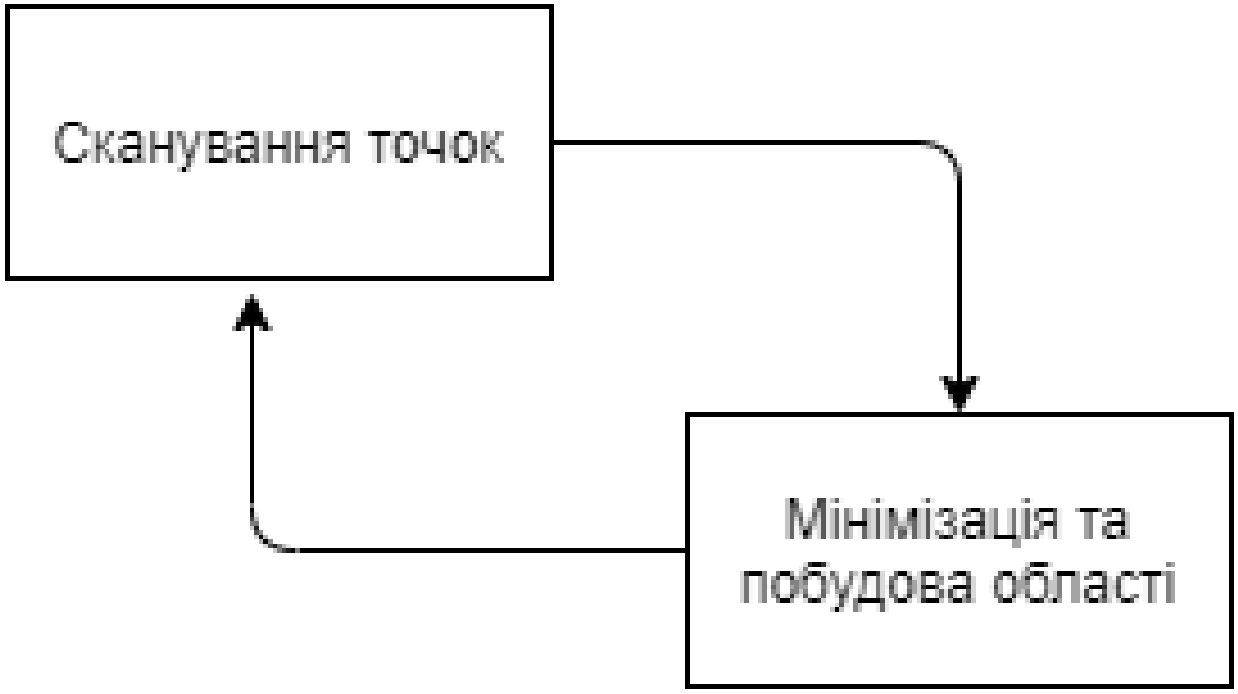
Додаток Є

Схематичне представлення роботи методу пошуку точок



Додаток Ж

Схематичне представлення роботи процесу побудови області

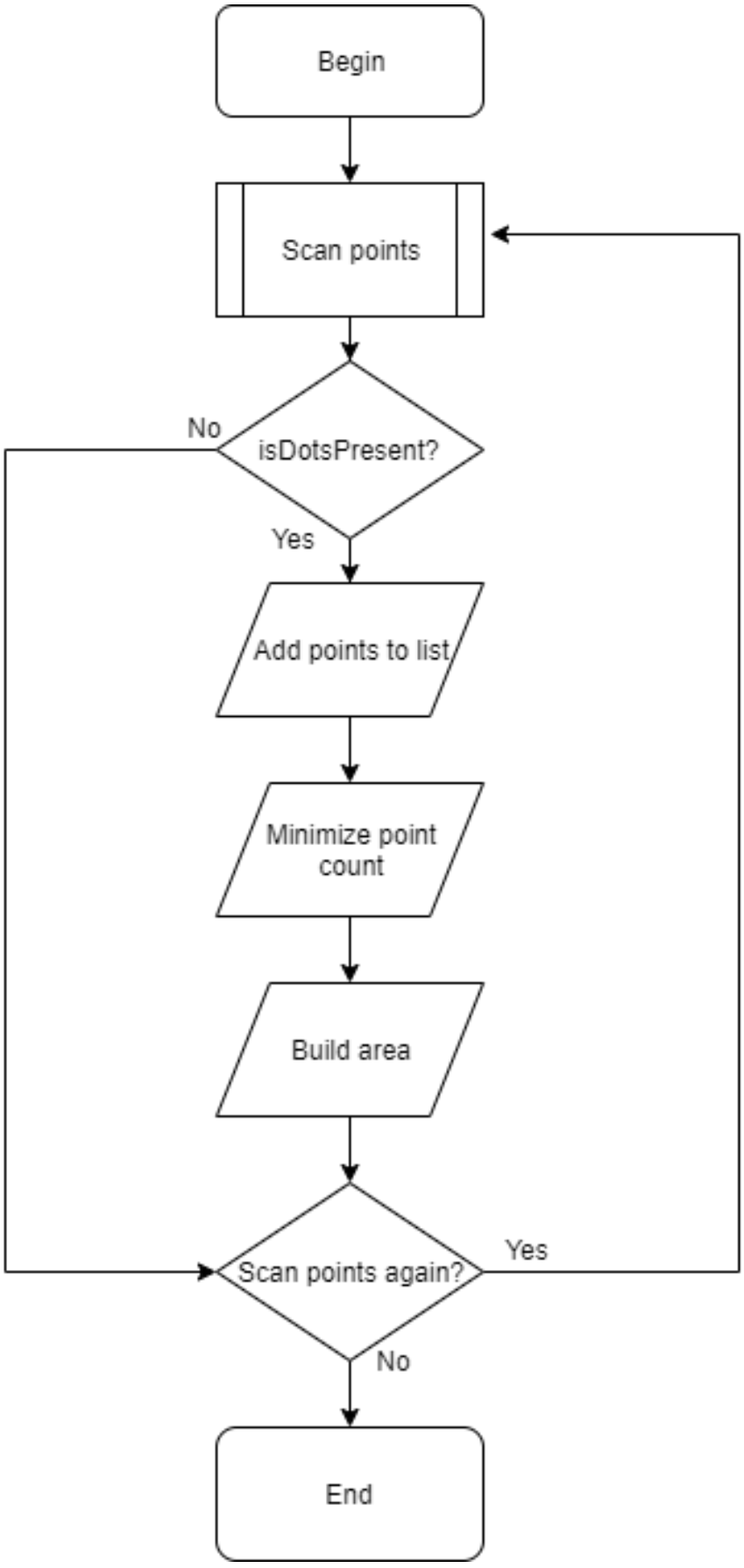


Додаток З

Схематичне доповнене представлення роботи методу сканування точок та побудова області

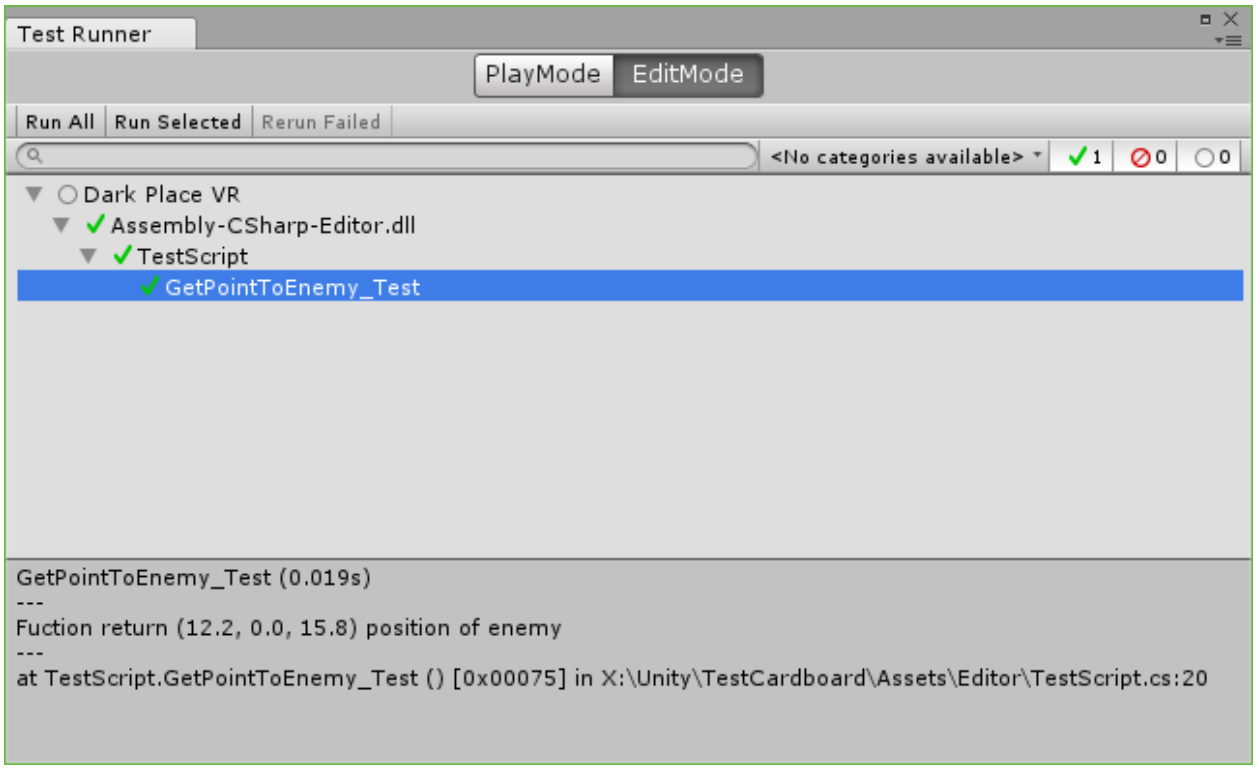
Додаток І

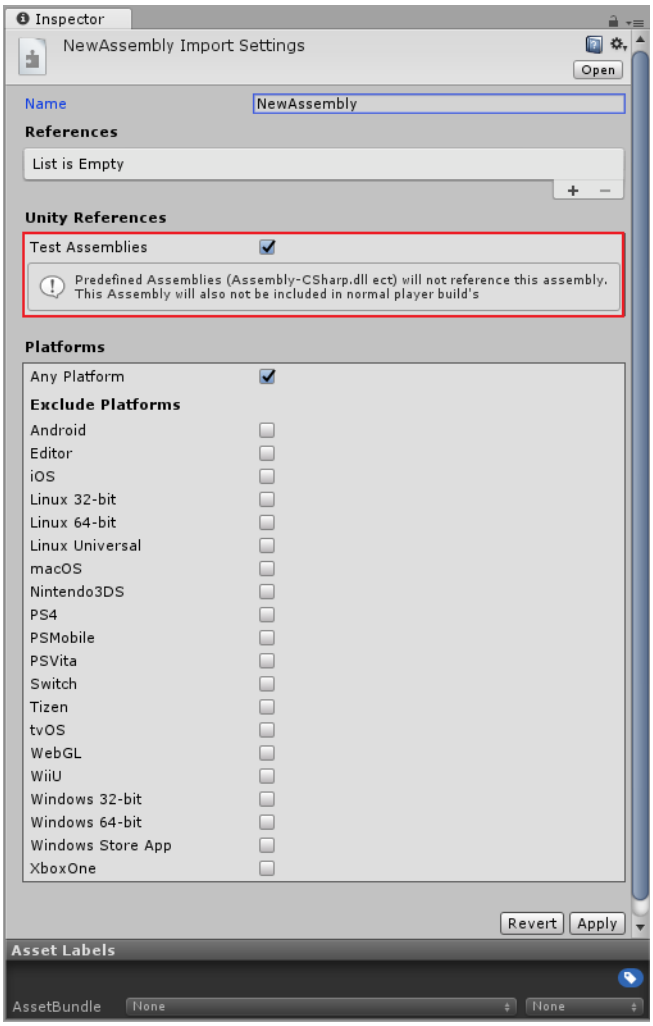
Блок схема роботи алгоритму


Begin
Scan points
No
isDotsPresent?
Yes
Add points to list
Minimize point count
Build area
Scan points again?
Yes
No
End

Додаток Й

Тестування і перевірка програми




Inspector
NewAssembly Import Settings
Open
Name
NewAssembly
References
List is Empty
Unity References
Test Assemblies
Predefined Assemblies (Assembly-CSharp.dll ect) will not reference this assembly.
This Assembly will also not be included in normal player build's
Platforms
Any Platform
Exclude Platforms
Android
Editor
iOS
Linux 32-bit
Linux 64-bit
Linux Universal
macOS
Nintendo3DS
PS4
PSMobile
PSVita
Switch
Tizen
tvOS
WebGL
WiiU
Windows 32-bit
Windows 64-bit
Windows Store App
XboxOne
Revert
Apply
Asset Labels
AssetBundle
None
None

Додаток К

Лістинг програми

```
    using GoogleARCore.Examples.ObjectManipulation;
    using System;
    using System.Collections;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using UnityEngine;
    using UnityEngine.Networking;
    using UnityEngine.UI;

    public class DownloadHelper : MonoBehaviour
    {

        public static DownloadHelper Instance { get; private
set; }

        [SerializeField] private string[] items;

        private Firebase.Storage.FirebaseStorage storage;
        private PawnManipulator andyPlacement;
        private UIController uiController;

        void Awake()
        {
            if (Instance == null)
            {
                Instance = this;
            }
            else if (Instance == this)
            {
                Destroy(gameObject);
            }
            DontDestroyOnLoad(gameObject);
            storage =
Firebase.Storage.FirebaseStorage.DefaultInstance;
        }

        void Start()
        {
            //CreateUserOnce();
            uiController = FindObjectOfType<UIController>();
```

```
            andyPlacement =
FindObjectOfType<PawnManipulator>();
            CreateButtonsForItems();
            //StartCoroutine(DownloadAssetBundle("beds",
selectedPrefabToDownload));
        }

        public void CreateButtonsForItems()
        {
            uiController.CreateButtonsForItems(items);
        }

        public IEnumerator DownloadAssetBundle(string
category, string prefabName)
        {
            while (!Caching.ready)
            {
                yield return null;
            }
            while (!FirebaseLoginIn.Instance.IsLoginIn)
            {
                yield return null;
            }
            WWW www = null;
            string url = string.Empty;
            storage =
Firebase.Storage.FirebaseStorage.DefaultInstance;
            Firebase.Storage.StorageReference
storageReference =
storage.GetReferenceFromUrl("gs://myarproject-
f4b82.appspot.com/" + prefabName);

storageReference.GetDownloadUrlAsync().ContinueWith((Task<Ur
i> task) =>
            {
                if (!task.IsFaulted && !task.IsCanceled)
                {
                    Debug.Log("Download url: " +
task.Result);
                    url = task.Result.ToString();
                }
                else
                {
                    Debug.Log(task.Result);
                }
```

```
            });
            while (string.IsNullOrEmpty(url))
            {
                yield return null;
            }
            www = WWW.LoadFromCacheOrDownload(url, 0);
            yield return www;
            if (!string.IsNullOrEmpty(www.error))
            {
                Debug.LogError(www.error);
                yield break;
            }
            Debug.Log("FILE WAS DOWNLOAD!");
            var assetBundle = www.assetBundle;
            var gameObj =
assetBundle.LoadAssetAsync(prefabName, typeof(GameObject));
            yield return gameObj;
            andyPlacement.PawnPrefab = gameObj.asset as
GameObject;
            assetBundle.Unload(false);
        }
    }
    //-----------------------------------------------------
----------------
    // <copyright file="PawnManipulator.cs" company="Google">
    //
    // Copyright 2019 Google LLC All Rights Reserved.
    //
    // Licensed under the Apache License, Version 2.0 (the
"License");
    // you may not use this file except in compliance with
the License.
    // You may obtain a copy of the License at
    //
    // http://www.apache.org/licenses/LICENSE-2.0
    //
    // Unless required by applicable law or agreed to in
writing, software
    // distributed under the License is distributed on an "AS
IS" BASIS,
    // WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied.
    // See the License for the specific language governing
permissions and
    // limitations under the License.
```

```
    //
    // </copyright>
    //-------------------------------------------------------
----------------

    namespace GoogleARCore.Examples.ObjectManipulation
    {
        using GoogleARCore;
        using System.Collections.Generic;
        using UnityEngine;
        using UnityEngine.EventSystems;

        /// <summary>
        /// Controls the placement of objects via a tap
gesture.
        /// </summary>
        public class PawnManipulator : Manipulator
        {
            /// <summary>
            /// The first-person camera being used to render
the passthrough camera image (i.e. AR
            /// background).
            /// </summary>
            public Camera FirstPersonCamera;

            /// <summary>
            /// A prefab to place when a raycast from a user
touch hits a plane.
            /// </summary>
            public GameObject PawnPrefab;

            /// <summary>
            /// Manipulator prefab to attach placed objects
to.
            /// </summary>
            public GameObject ManipulatorPrefab;

            /// <summary>
            /// Returns true if the manipulation can be
started for the given gesture.
            /// </summary>
            /// <param name="gesture">The current
gesture.</param>
            /// <returns>True if the manipulation can be
started.</returns>
```

```
            protected override bool
CanStartManipulationForGesture(TapGesture gesture)
            {
                if (gesture.TargetObject == null)
                {
                    return true;
                }

                return false;
            }

            /// <summary>
            /// Function called when the manipulation is
ended.
            /// </summary>
            /// <param name="gesture">The current
gesture.</param>
            protected override void
OnEndManipulation(TapGesture gesture)
            {
                if(IsPointerOverUIObject())
                {
                    return;
                }

                if (gesture.WasCancelled)
                {
                    return;
                }

                // If gesture is targeting an existing
object, we are done.
                if (gesture.TargetObject != null)
                {
                    return;
                }

                // Raycast against the location the player
touched to search for planes.
                TrackableHit hit;
                TrackableHitFlags raycastFilter =
TrackableHitFlags.PlaneWithinPolygon;

                if (Frame.Raycast(gesture.StartPosition.x,
gesture.StartPosition.y, raycastFilter, out hit))
```

```
                {
                    // Use hit pose and camera pose to check
if hit test is from the
                    // back of the plane, if it is, no need
to create the anchor.
                    if ((hit.Trackable is DetectedPlane) &&

Vector3.Dot(FirstPersonCamera.transform.position -
hit.Pose.position,
                            hit.Pose.rotation * Vector3.up) <
0)
                    {
                        Debug.Log("Hit at back of the current
DetectedPlane");
                    }
                    else
                    {
                        // Instantiate Andy model at the hit
pose.
                        var andyObject =
Instantiate(PawnPrefab, hit.Pose.position,
hit.Pose.rotation);

                        // Instantiate manipulator.
                        var manipulator =
Instantiate(ManipulatorPrefab, hit.Pose.position,
hit.Pose.rotation);

                        // Make Andy model a child of the
manipulator.
                        andyObject.transform.parent =
manipulator.transform;

                        // Create an anchor to allow ARCore
to track the hitpoint as understanding of the physical
                        // world evolves.
                        var anchor =
hit.Trackable.CreateAnchor(hit.Pose);

                        // Make manipulator a child of the
anchor.
                        manipulator.transform.parent =
anchor.transform;

                        // Select the placed object.
```

```
manipulator.GetComponent<Manipulator>().Select();
                    }
                }
            }

            /// <summary>
            /// Cast a ray to test if Input.mousePosition is
over any UI object in EventSystem.current. This is a
replacement
            /// for IsPointerOverGameObject() which does not
work on Android in 4.6.0f3
            /// </summary>
            private bool IsPointerOverUIObject()
            {
                PointerEventData eventDataCurrentPosition =
new PointerEventData(EventSystem.current);
                eventDataCurrentPosition.position = new
Vector2(Input.mousePosition.x, Input.mousePosition.y);

                List<RaycastResult> results = new
List<RaycastResult>();

EventSystem.current.RaycastAll(eventDataCurrentPosition,
results);
                return results.Count > 0;
            }
        }
    }
    //-------------------------------------------------------
----------------
    // <copyright file="ManipulationSystem.cs"
company="Google">
    //
    // Copyright 2018 Google Inc. All Rights Reserved.
    //
    // Licensed under the Apache License, Version 2.0 (the
"License");
    // you may not use this file except in compliance with
the License.
    // You may obtain a copy of the License at
    //
    // http://www.apache.org/licenses/LICENSE-2.0
    //
```

```
    // Unless required by applicable law or agreed to in
writing, software
    // distributed under the License is distributed on an "AS
IS" BASIS,
    // WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied.
    // See the License for the specific language governing
permissions and
    // limitations under the License.
    //
    // </copyright>
    //-------------------------------------------------------
----------------

    namespace GoogleARCore.Examples.ObjectManipulation
    {
        using UnityEngine;

        /// <summary>
        /// Manipulation system allows the user to manipulate
virtual objects (select, translate,
        /// rotate, scale and elevate) through gestures (tap,
drag, twist, swipe).
        /// Manipulation system also handles the current
selected object and its visualization.
        ///
        /// To enable it add one ManipulationSystem to your
scene and one Manipulator as parent of each
        /// of your virtual objects.
        /// </summary>
        public class ManipulationSystem : MonoBehaviour
        {
            private static ManipulationSystem s_Instance =
null;

            private DragGestureRecognizer
m_DragGestureRecognizer = new DragGestureRecognizer();

            private PinchGestureRecognizer
m_PinchGestureRecognizer = new PinchGestureRecognizer();

            private TwoFingerDragGestureRecognizer
m_TwoFingerDragGestureRecognizer = new
TwoFingerDragGestureRecognizer();
```

```
            private TapGestureRecognizer
m_TapGestureRecognizer = new TapGestureRecognizer();

            private TwistGestureRecognizer
m_TwistGestureRecognizer = new TwistGestureRecognizer();

            [SerializeField] private UIController
uiController;

            void Start()
            {
                uiController =
FindObjectOfType<UIController>();
            }

            /// <summary>
            /// Gets the ManipulationSystem instance.
            /// </summary>
            public static ManipulationSystem Instance
            {
                get
                {
                    if (s_Instance == null)
                    {
                        var manipulationSystems =
FindObjectsOfType<ManipulationSystem>();
                        if (manipulationSystems.Length > 0)
                        {
                            s_Instance =
manipulationSystems[0];
                        }
                        else
                        {
                            Debug.LogError("No instance of
ManipulationSystem exists in the scene.");
                        }
                    }

                    return s_Instance;
                }
            }

            /// <summary>
            /// Gets the Drag gesture recognizer.
            /// </summary>
```

```
            public DragGestureRecognizer
DragGestureRecognizer
            {
                get
                {
                    return m_DragGestureRecognizer;
                }
            }

            /// <summary>
            /// Gets the Pinch gesture recognizer.
            /// </summary>
            public PinchGestureRecognizer
PinchGestureRecognizer
            {
                get
                {
                    return m_PinchGestureRecognizer;
                }
            }

            /// <summary>
            /// Gets the two finger drag gesture recognizer.
            /// </summary>
            public TwoFingerDragGestureRecognizer
TwoFingerDragGestureRecognizer
            {
                get
                {
                    return m_TwoFingerDragGestureRecognizer;
                }
            }

            /// <summary>
            /// Gets the Tap gesture recognizer.
            /// </summary>
            public TapGestureRecognizer TapGestureRecognizer
            {
                get
                {
                    return m_TapGestureRecognizer;
                }
            }

            /// <summary>
```

```
            /// Gets the Twist gesture recognizer.
            /// </summary>
            public TwistGestureRecognizer
TwistGestureRecognizer
            {
                get
                {
                    return m_TwistGestureRecognizer;
                }
            }

            /// <summary>
            /// Gets the current selected object.
            /// </summary>
            public GameObject SelectedObject { get; private
set; }

            /// <summary>
            /// The Unity Awake() method.
            /// </summary>
            public void Awake()
            {
                if (Instance != this)
                {
                    Debug.LogWarning("Multiple instances of
ManipulationSystem detected in the scene." +
                                     " Only one instance can
exist at a time. The duplicate instances" +
                                     " will be destroyed.");
                    DestroyImmediate(gameObject);
                    return;
                }

                DontDestroyOnLoad(gameObject);
            }

            /// <summary>
            /// The Unity Update() method.
            /// </summary>
            public void Update()
            {
                DragGestureRecognizer.Update();
                PinchGestureRecognizer.Update();
                TwoFingerDragGestureRecognizer.Update();
                TapGestureRecognizer.Update();
```

```
                TwistGestureRecognizer.Update();
            }

            /// <summary>
            /// Deselects the selected object if any.
            /// </summary>
            internal void Deselect()
            {
                SelectedObject = null;
            }

            /// <summary>
            /// Select an object.
            /// </summary>
            /// <param name="target">The object to
select.</param>
            internal void Select(GameObject target)
            {
                if (SelectedObject == target)
                {
                    return;
                }

                Deselect();
                SelectedObject = target;
                Debug.LogError("Selected");

uiController.SetSelectedOnject(target.GetComponentInChildren
<ObjectMaterialChanger>().gameObject);

StartCoroutine(uiController.CreateButtonsWithDelay(.1f));
            }
        }
    }
    using System;
    using System.Collections;
    using System.Collections.Generic;
    using UnityEngine;
    using UnityEngine.UI;

    public class UIController : MonoBehaviour
    {

        public GameObject SelectedObject { get; set; }
```

```
        [Header("Objects spawn"), SerializeField] private
Transform buttonsParent;
        [SerializeField] private Button objectButtonPrefab;
        [Header("Materials spawn"), SerializeField] private
Transform materialsParent;
        [SerializeField] private Button
materialsButtonPrefab;
        [Header("Panel"), SerializeField] private GameObject
materialsPanel, buttonPanel;
        [SerializeField] private Button toItemsButton;

        private Animator materialsAnimator, buttonAnimator;

        void Start()
        {
            materialsAnimator =
materialsPanel.GetComponent<Animator>();
            buttonAnimator =
buttonPanel.GetComponent<Animator>();
            toItemsButton.onClick.AddListener(() => {
DownloadHelper.Instance.CreateButtonsForItems(); });
        }

        public void SetSelectedOnject(GameObject
selectedObject)
        {
            SelectedObject = selectedObject;
            Debug.Log("Selected: " + SelectedObject.name);
        }

        public IEnumerator CreateButtonsWithDelay(float time
= 1f)
        {
            yield return new WaitForSeconds(time);
            CreateButtons();
        }

        public void CreateButtons()
        {
            if (SelectedObject == null)
            {
                Debug.LogError("Selected object is null,
please set gameObject to this field");
                return;
            }
```

```
            buttonAnimator.SetTrigger("CloseUIElement");
            materialsAnimator.SetTrigger("CloseUIElement");
            var objects =
SelectedObject.GetComponent<ObjectMaterialChanger>().GetObje
ctList();
            Debug.Log(objects.Count);
            DeleteAnotherObjects(buttonsParent);
            foreach (var obj in objects)
            {
                var objectButton =
Instantiate(objectButtonPrefab, buttonsParent);

objectButton.GetComponentInChildren<Text>().text = obj.name;
                objectButton.onClick.AddListener(() =>
                {
                    DeleteAnotherObjects(materialsParent);
                    CreateMaterials(obj);
                    Debug.Log("Selected " + obj.name);
                });
            }
            buttonAnimator.SetTrigger("OpenUIElement");
            materialsAnimator.SetTrigger("OpenUIElement");
        }

        public void CreateButtonsForItems(string[] items)
        {
            buttonAnimator.SetTrigger("CloseUIElement");
            materialsAnimator.SetTrigger("CloseUIElement");
            DeleteAnotherObjects(buttonsParent);
            DeleteAnotherObjects(materialsParent);
            foreach (var item in items)
            {
                var itemButton =
Instantiate(objectButtonPrefab, buttonsParent);

itemButton.GetComponentInChildren<Text>().text = item;
                itemButton.onClick.AddListener(() =>
                {

StartCoroutine(DownloadHelper.Instance.DownloadAssetBundle(s
tring.Empty, item));
                });
            }
            buttonAnimator.SetTrigger("OpenUIElement");
```

```
        }

        private void DeleteAnotherObjects(Transform
materialsParent)
        {
            foreach (Transform child in materialsParent)
            {
                Destroy(child.gameObject);
            }
        }

        private void CreateMaterials(ObjElements obj)
        {
            var materials = obj.GetAllElementMaterials();
            foreach (var mat in materials)
            {
                var materialButton =
Instantiate(objectButtonPrefab, materialsParent);

materialButton.GetComponentInChildren<Text>().text = "";

materialButton.GetComponentInChildren<Image>().sprite =
Sprite.Create((Texture2D)mat.mainTexture, new Rect(0.0f,
0.0f, mat.mainTexture.width, mat.mainTexture.height), new
Vector2(0.5f, 0.5f), 100.0f);
                materialButton.onClick.AddListener(() =>
                {
                    Material currentMaterial = mat;
                    obj.Element.material = currentMaterial;
                    Debug.Log("Selected material: " +
mat.name);
                });
            }
            materialsAnimator.SetTrigger("OpenUIElement");
        }
    }
    using System.Collections;
    using System.Collections.Generic;
    using UnityEngine;

    [System.Serializable]
    public class ObjElements : MonoBehaviour
    {
        public Renderer Element { get; private set; }
```

```
        [SerializeField] private List<Material>
ElementMaterials;

        void Start()
        {
            Element = GetComponent<Renderer>();
        }

        public List<Material> GetAllElementMaterials()
        {
            return ElementMaterials;
        }
    }
    using System.Collections;
    using System.Collections.Generic;
    using System.Linq;
    using UnityEngine;

    public class ObjectMaterialChanger : MonoBehaviour
    {

        [SerializeField] private List<ObjElements>
objectElements;

        void Start()
        {
            Init();
        }

        private void Init()
        {
            objectElements =
GetComponentsInChildren<ObjElements>().ToList();
        }

        public List<ObjElements> GetObjectList()
        {
            return objectElements;
        }
    }
    using System.Collections;
    using System.Collections.Generic;
    using System.IO;
    using System.Text;
    using UnityEngine;
```

```
    public class FirebaseLoginIn : MonoBehaviour
    {

        public static FirebaseLoginIn Instance { get; private
set; }

        [SerializeField] private string email;
        [SerializeField] private string password;

        public bool IsLoginIn { get; private set; } = false;
        private int version = 0;

        private Firebase.Auth.FirebaseAuth auth;

        void Awake()
        {
            if (Instance == null)
            {
                Instance = this;
            }
            else if (Instance == this)
            {
                Destroy(gameObject);
            }
            DontDestroyOnLoad(gameObject);

            Init();
        }

        private void Init()
        {
            auth =
Firebase.Auth.FirebaseAuth.DefaultInstance;
        }

        // Use this for initialization
        void Start()
        {
            LoginToFirebase();
        }

        private void CreateUserOnce()
        {
```

```
            auth.CreateUserWithEmailAndPasswordAsync(email,
password).ContinueWith(task =>
            {
                if (task.IsCanceled)
                {

Debug.LogError("CreateUserWithEmailAndPasswordAsync was
canceled.");
                    return;
                }
                if (task.IsFaulted)
                {

Debug.LogError("CreateUserWithEmailAndPasswordAsync
encountered an error: " + task.Exception);
                    return;
                }

                // Firebase user has been created.
                Firebase.Auth.FirebaseUser newUser =
task.Result;
                Debug.LogFormat("Firebase user created
successfully: {0} ({1})",
                    newUser.DisplayName, newUser.UserId);
            });
        }

        private void LoginToFirebase()
        {
            Firebase.Auth.Credential credential =

Firebase.Auth.EmailAuthProvider.GetCredential(email,
password);

auth.SignInWithCredentialAsync(credential).ContinueWith(task
=>
            {
                if (task.IsCanceled)
                {
                    Debug.LogError("SignInWithCredentialAsync
was canceled.");
                    CreateUserOnce();
                    LoginToFirebase();
                    return;
                }
```

```
                if (task.IsFaulted)
                {
                    Debug.LogError("SignInWithCredentialAsync
encountered an error: " + task.Exception);
                    CreateUserOnce();
                    LoginToFirebase();
                    return;
                }

                Firebase.Auth.FirebaseUser newUser =
task.Result;
                Debug.LogFormat("User signed in successfully:
{0} ({1})",
                    newUser.DisplayName, newUser.UserId);

                IsLoginIn = true;
            });
        }
    }
    //----------------------------------------------------------
----------------
    // <copyright file="Anchor.cs" company="Google">
    //
    // Copyright 2017 Google Inc. All Rights Reserved.
    //
    // Licensed under the Apache License, Version 2.0 (the
"License");
    // you may not use this file except in compliance with
the License.
    // You may obtain a copy of the License at
    //
    // http://www.apache.org/licenses/LICENSE-2.0
    //
    // Unless required by applicable law or agreed to in
writing, software
    // distributed under the License is distributed on an "AS
IS" BASIS,
    // WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied.
    // See the License for the specific language governing
permissions and
    // limitations under the License.
    //
    // </copyright>
```

```
    //-------------------------------------------------------
----------------

    namespace GoogleARCore
    {
        using System;
        using System.Collections.Generic;
        using System.Diagnostics.CodeAnalysis;
        using GoogleARCoreInternal;
        using UnityEngine;

        /// <summary>
        /// Attaches a GameObject to an ARCore <see
cref="Trackable"/>.  The transform of the GameObject will
        /// be updated to maintain the semantics of the
attachment relationship, which varies between
        /// sub-types of Trackable.
        /// </summary>

[HelpURL("https://developers.google.com/ar/reference/unity/c
lass/GoogleARCore/Anchor")]
        public class Anchor : MonoBehaviour
        {
            private static Dictionary<IntPtr, Anchor>
s_AnchorDict =
                new Dictionary<IntPtr, Anchor>(new
IntPtrEqualityComparer());

            private TrackingState m_LastFrameTrackingState =
TrackingState.Stopped;

            private bool m_IsSessionDestroyed = false;

            /// <summary>
            /// Gets the tracking state of the anchor.
            /// </summary>
            public TrackingState TrackingState
            {
                get
                {
                    if (_IsSessionDestroyed())
                    {
                        // Anchors from another session are
considered stopped.
                        return TrackingState.Stopped;
```

```
                }

                return
NativeSession.AnchorApi.GetTrackingState(NativeHandle);
            }
        }

        internal NativeSession NativeSession { get;
private set; }

        internal IntPtr NativeHandle { get; private set;
}

        internal TrackableApi TrackableApi
        {
            get
            {
                return default(TrackableApi);
            }

            set
            {
            }
        }

        internal static Anchor Factory(NativeSession
nativeApi, IntPtr anchorNativeHandle,
            bool isCreate = true)
        {
            if (anchorNativeHandle == IntPtr.Zero)
            {
                return null;
            }

            Anchor result;
            if
(s_AnchorDict.TryGetValue(anchorNativeHandle, out result))
            {
                // Release acquired handle and return
cached result
                AnchorApi.Release(anchorNativeHandle);
                return result;
            }

            if (isCreate)
```

```
                {
                    Anchor anchor = (new
GameObject()).AddComponent<Anchor>();
                    anchor.gameObject.name = "Anchor";
                    anchor.NativeHandle = anchorNativeHandle;
                    anchor.NativeSession = nativeApi;
                    anchor.Update();

                    s_AnchorDict.Add(anchorNativeHandle,
anchor);
                    return anchor;
                }

                return null;
            }

            /// <summary>
            /// The Unity Update method.
            /// </summary>
            private void Update()
            {
                if (NativeHandle == IntPtr.Zero)
                {
                    Debug.LogError(
                        "Anchor components instantiated
outside of ARCore are not supported. " +
                        "Please use a 'Create' method within
ARCore to instantiate anchors.");
                    return;
                }

                if (_IsSessionDestroyed())
                {
                    return;
                }

                var pose =
NativeSession.AnchorApi.GetPose(NativeHandle);
                transform.position = pose.position;
                transform.rotation = pose.rotation;

                TrackingState currentFrameTrackingState =
TrackingState;
                if (m_LastFrameTrackingState !=
currentFrameTrackingState)
```

```
                {
                    bool isAnchorTracking =
currentFrameTrackingState == TrackingState.Tracking;
                    foreach (Transform child in transform)
                    {

child.gameObject.SetActive(isAnchorTracking);
                    }

                    m_LastFrameTrackingState =
currentFrameTrackingState;
                }
            }

            private void OnDestroy()
            {
                if (NativeHandle == IntPtr.Zero)
                {
                    return;
                }

                if (NativeSession != null &&
!NativeSession.IsDestroyed)
                {

NativeSession.AnchorApi.Detach(NativeHandle);
                }

                s_AnchorDict.Remove(NativeHandle);
                AnchorApi.Release(NativeHandle);
            }

            private bool _IsSessionDestroyed()
            {
                if (!m_IsSessionDestroyed)
                {
                    var nativeSession =
LifecycleManager.Instance.NativeSession;
                    if (nativeSession != NativeSession)
                    {
                        Debug.LogErrorFormat(
                            "The session which created this
anchor has been destroyed. " +
                            "The anchor on GameObject {0} can
no longer update.",
```

```
                            this.gameObject != null ?
this.gameObject.name : "Unknown");
                        m_IsSessionDestroyed = true;
                    }
                }

                return m_IsSessionDestroyed;
            }
        }
    }
    //-------------------------------------------------------
----------------
    // <copyright file="TrackableApi.cs" company="Google">
    //
    // Copyright 2017 Google Inc. All Rights Reserved.
    //
    // Licensed under the Apache License, Version 2.0 (the
"License");
    // you may not use this file except in compliance with
the License.
    // You may obtain a copy of the License at
    //
    // http://www.apache.org/licenses/LICENSE-2.0
    //
    // Unless required by applicable law or agreed to in
writing, software
    // distributed under the License is distributed on an "AS
IS" BASIS,
    // WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied.
    // See the License for the specific language governing
permissions and
    // limitations under the License.
    //
    // </copyright>
    //-------------------------------------------------------
----------------

    namespace GoogleARCoreInternal
    {
        using System;
        using System.Collections.Generic;
        using GoogleARCore;
        using UnityEngine;
```

```
    #if UNITY_IOS && !UNITY_EDITOR
        using AndroidImport =
GoogleARCoreInternal.DllImportNoop;
        using IOSImport =
System.Runtime.InteropServices.DllImportAttribute;
    #else
        using AndroidImport =
System.Runtime.InteropServices.DllImportAttribute;
        using IOSImport = GoogleARCoreInternal.DllImportNoop;
    #endif

        internal class TrackableApi
        {
            private NativeSession m_NativeSession;

            public TrackableApi(NativeSession nativeSession)
            {
                m_NativeSession = nativeSession;
            }

            public ApiTrackableType GetType(IntPtr
trackableHandle)
            {
                ApiTrackableType type =
ApiTrackableType.Plane;

ExternApi.ArTrackable_getType(m_NativeSession.SessionHandle,
trackableHandle, ref type);
                return type;
            }

            public TrackingState GetTrackingState(IntPtr
trackableHandle)
            {
                ApiTrackingState apiTrackingState =
ApiTrackingState.Stopped;

ExternApi.ArTrackable_getTrackingState(m_NativeSession.Sessi
onHandle, trackableHandle,
                    ref apiTrackingState);
                return apiTrackingState.ToTrackingState();
            }

            public bool AcquireNewAnchor(IntPtr
trackableHandle, Pose pose, out IntPtr anchorHandle)
```

```
            {
                IntPtr poseHandle =
m_NativeSession.PoseApi.Create(pose);
                anchorHandle = IntPtr.Zero;
                int status =
ExternApi.ArTrackable_acquireNewAnchor(
                    m_NativeSession.SessionHandle,
trackableHandle, poseHandle, ref anchorHandle);
                m_NativeSession.PoseApi.Destroy(poseHandle);
                return status == 0;
            }

            public void Release(IntPtr trackableHandle)
            {

ExternApi.ArTrackable_release(trackableHandle);
            }

            public void GetAnchors(IntPtr trackableHandle,
List<Anchor> anchors)
            {
                IntPtr anchorListHandle =
m_NativeSession.AnchorApi.CreateList();
                ExternApi.ArTrackable_getAnchors(
                    m_NativeSession.SessionHandle,
trackableHandle, anchorListHandle);

                anchors.Clear();
                int anchorCount =
m_NativeSession.AnchorApi.GetListSize(anchorListHandle);
                for (int i = 0; i < anchorCount; i++)
                {
                    IntPtr anchorHandle =

m_NativeSession.AnchorApi.AcquireListItem(anchorListHandle,
i);
                    Anchor anchor =
Anchor.Factory(m_NativeSession, anchorHandle, false);
                    if (anchor == null)
                    {
                        Debug.LogFormat("Unable to find
Anchor component for handle {0}", anchorHandle);
                    }
                    else
                    {
```

```
                    anchors.Add(anchor);
                }
            }


m_NativeSession.AnchorApi.DestroyList(anchorListHandle);
        }

        private struct ExternApi
        {
    #pragma warning disable 626
            [AndroidImport(ApiConstants.ARCoreNativeApi)]
            public static extern void
ArTrackable_getType(
                IntPtr sessionHandle, IntPtr
trackableHandle, ref ApiTrackableType trackableType);

            [AndroidImport(ApiConstants.ARCoreNativeApi)]
            public static extern void
ArTrackable_getTrackingState(
                IntPtr sessionHandle, IntPtr
trackableHandle, ref ApiTrackingState trackingState);

            [AndroidImport(ApiConstants.ARCoreNativeApi)]
            public static extern int
ArTrackable_acquireNewAnchor(
                IntPtr sessionHandle, IntPtr
trackableHandle, IntPtr poseHandle,
                ref IntPtr anchorHandle);

            [AndroidImport(ApiConstants.ARCoreNativeApi)]
            public static extern void
ArTrackable_release(IntPtr trackableHandle);

            [AndroidImport(ApiConstants.ARCoreNativeApi)]
            public static extern void
ArTrackable_getAnchors(
                IntPtr sessionHandle, IntPtr
trackableHandle, IntPtr outputListHandle);
    #pragma warning restore 626
        }
    }
```